2007-05-07

Fault Location for Power Transmission Systems Using Magnetic Field Sensing Coils

Kurt Josef Ferreira
Worcester Polytechnic Institute

Fault Location for Power Transmission Systems Using Magnetic Field Sensing Coils

by

Kurt Josef Ferreira

A Thesis

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Master of Science

in

Electrical and Computer Engineering

by

April 2007

Approved:

_____

Prof. Alexander E. Emanuel, Major Advisor

_____

Prof. James W. Matthews, Thesis Committee Member

_____

Dr. Herbert M. Pflanz, Thesis Committee Member

Abstract

The detection and location of faults on power transmission lines is essential to the protection and maintenance of a power system. Most methods of fault detection and location rely on measurements of electrical quantities provided by current and voltage transformers. These transformers can be expensive and require physical contact with the monitored high voltage equipment.

In this work, current transformers were replaced by magnetic field sensing coils. Such coils can be located remotely from substations and switching stations and do not require physical contact with the conductors. Rather than observing each individual conductor, the use of the magnetic field sensors allows the monitoring of the transmission line condition using a collective quantity. This study explores the use of the magnetic field sensors as an alternative measurement device for fault detection and location.

Acknowledgements

I would like to thank my advisor, Prof. Alexander E. Emanuel, for his teaching, guidance, and especially his patience throughout the past two years. I would also like to thank Prof. James W. Matthews and Dr. Herbert M. Pflanz for being willing to serve as my thesis committee members.

Thanks also to the ECE Department of Worcester Polytechnic Institute for providing me so many opportunities as both an undergraduate student and a graduate student.

Additionally, I would like to thank the Independent System Operator of New England (ISO-NE) for their funding and support during my graduate studies.

Finally, I would like to express my gratitude to my family and friends who have been so supportive and understanding throughout my continuing academic journey.

Kurt Ferreira
May 2007

Contents

List of Figures

List of Symbols

H_x	Total horizontal magnetic field intensity
$H_{x,k}$	Horizontal magnetic field intensity due to conductor k
H_y	Total vertical magnetic field intensity
$H_{y,k}$	Vertical magnetic field intensity due to conductor k
D_k	Horizontal magnetic field position constant for conductor k
Q_k	Vertical magnetic field position constant for conductor k
i_k	Current in conductor k
I_k	RMS current in conductor k
γ_k	Angle from vertical between conductor k and the magnetic field sensor
ℓ_k	Distance between conductor k and the magnetic field sensor
h_k	Vertical distance between conductor k and the magnetic field sensor
L_k	Horizontal distance between conductor k and the magnetic field sensor
X_k	Coefficient of sine function in conversion of a phase-shifted sinusoidal current in conductor k into a sum of a sine and cosine
Y_k	Coefficient of cosine function in conversion of a phase-shifted sinusoidal current in conductor k into a sum of a sine and cosine
t	Time
ω	Fundamental frequency
φ_k	Phase shift in the sinusoidal current in conductor k
[XY]	Sine and cosine coefficient matrix – columns of X and Y coefficients
[P]	Position matrix – rows of D and Q coefficients

$\hat{H}_x$	Magnitude of horizontal magnetic field
$\hat{H}_y$	Magnitude of vertical magnetic field
ϕ_x	Phase shift in sinusoidal representation of horizontal magnetic field
ϕ_y	Phase shift in sinusoidal representation of vertical magnetic field
[H]	Matrix of vertical and horizontal magnetic fields
A_x	Coefficient of sine function in conversion of a phase-shifted sinusoidal magnetic field in the horizontal direction into a sum of a sine and cosine
A_y	Coefficient of sine function in conversion of a phase-shifted sinusoidal magnetic field in the vertical direction into a sum of a sine and cosine
B_x	Coefficient of cosine function in conversion of a phase-shifted sinusoidal magnetic field in the horizontal direction into a sum of a sine and cosine
B_y	Coefficient of cosine function in conversion of a phase-shifted sinusoidal magnetic field in the vertical direction into a sum of a sine and cosine
r	Magnitude of vertical magnetic field divided by magnitude of horizontal magnetic field; used to simplify equations
ϕ	Difference between phase shift of vertical magnetic field and phase shift of horizontal magnetic field
α	Angle of rotation of the magnetic field ellipse
$\hat{H}_a$	Magnitude of one half-axis of the magnetic field
$\hat{H}_b$	Magnitude of the other half-axis of the magnetic field
p	Relative distance between the conductors and sensors
v_x	Voltage induced in the horizontal search coil
v_y	Voltage induced in the vertical search coil

μ_0	Permeability of free space ($4\pi \times 10^{-7}$ H/m)
N	Number of turns in each search coil
A	Equivalent area of each search coil
K	Coil constant of each search coil
L	Inductance of the transmission line per unit length
C	Capacitance of the transmission line per unit length
u	Propagation velocity in the transmission line
l	Total length of the transmission line
t_{trans}	Time for a pulse to travel from one end of the transmission line to the other end
d_1	Distance from a fault to the first end of the transmission line
d_2	Distance from a fault to the other end of the transmission line
t_1	Time at which a fault is detected at the first end of the transmission line
t_2	Time at which a fault is detected at the other end of the transmission line
t_{fault}	Time at which a fault actually occurs
Δ_{step}	Fault location step size due to sampling rate
SR	Sampling rate of the analog to digital converter used for analysis
ρ	Distance from the origin of a point in polar coordinates (radial coordinate)
ρ_{min}	Minimum distance from the origin in the elliptical rotating magnetic field
ρ_{max}	Maximum distance from the origin in the elliptical rotating magnetic field
θ	Angular coordinate of a point in polar coordinates
θ_{shift}	Angular rotation of the elliptical rotating magnetic field
Δ_ρ	Change in rho (ρ) for a set time step

ρ_t	Distance from the origin of the point at time t
Δ_θ	Change in theta (θ) for a set time step
θ_t	Angular coordinate at time t
d_{read}	Fault location detected by the algorithm
d_{actual}	Actual fault location

Summary

A variety of methods of detecting and locating faults on power transmission lines exist. Most of these methods utilize the measurements from voltage and current transformers at substations or switching stations to perform their analyses. This thesis examines the effectiveness of using magnetic field sensing coils as alternative measurement devices for the purpose of fault detection and location.

A review of common methods of fault location is presented. This review is focused on impedance-based and traveling wave-based fault location as they are the most common traditional methods. A few previous uses of magnetic field sensing coils in fault detection and location schemes are also discussed in order to determine the previously recognized benefits of using such coils.

The underlying mathematics used in determining the magnetic field due to an unspecified number of conductors and due to a three-conductor system are then examined. The results of this analysis are used in simulating the magnetic field for a variety of conductor configurations under normal operating conditions and for line to ground and line to line fault conditions. This information is used to determine the potential effectiveness of monitoring the magnetic field to detect faults.

Based on these findings, four algorithms are constructed which monitor the magnetic field near the transmission line for the purpose of fault detection. Each of these algorithms determines some aspect of the steady-state behavior of the magnetic field and attempts to detect any deviations from this behavior. These algorithms are described in detail and their comparative benefits and drawbacks are determined.

An implementation of a complete fault detection and location procedure which uses these algorithms in conjunction with one another is then described. This implementation is then used to test the combined effectiveness of the algorithms for a variety of fault types and fault resistances. The fault location errors for these tests are then presented. This information is used in determining the effectiveness of the magnetic field sensor as a measurement device for the purpose of fault detection and location.

1. Introduction and Description of Problem

Fault detection and location has been a goal of power system engineers since the creation of distribution and transmission systems. Quick fault detection can help protect equipment by allowing the disconnection of faulted lines before any significant damage is done. Accurate fault location can help utility personnel remove persistent faults and locate areas where faults regularly occur, thus reducing the frequency and length of power outages. As a result, while fault detection and location schemes have been developed in the past, a variety of algorithms continue to be developed to perform this task more accurately and more effectively [8-10].

Most analysis methods rely on the values of either current or voltage phasors measured by means of current or voltage transformers at substations or switching stations. To gather this information, at least three transformers are typically required at each end of the subtransmission or transmission line. These transducers are expensive, especially when the system involves high voltage lines. Some algorithms – particularly fault impedance-based algorithms – require both current and voltage information. However, it is possible to monitor a transmission system without using current or voltage transformers through the analysis of the magnetic field near the conductors.

Since each conductor in a transmission line creates a magnetic field due to the current through it, there is the possibility of analyzing the transmission line system based on the resultant magnetic field produced by its conductors. The magnetic detection is performed using two sensing coils at each end of the transmission line. One detects the vertical magnetic field intensity and the other detects the horizontal magnetic field

intensity. The two-dimensional magnetic field intensity can then be resolved from this information.

Just as with many other fault detection methods, faults are detected when unexpected changes occur within the monitored data. The only difference is that this analysis attempts to detect changes in the vertical and horizontal magnetic field intensities rather than individual changes in the monitored voltages or conductor currents. The fault detection and location method discussed in this thesis is based on analysis of the expected behavior of the magnetic field near a transmission line and algorithms for detecting unexpected variations.

2. Background

Fault detection is essential to the safe operation of electric power transmission and distribution systems. Without some sort of fault detection, the automated removal of short circuits from a transmission system would be impossible. As a result, these faults might persist until essential electrical equipment is damaged or destroyed. Fault location is not necessarily essential to power system protection, but it can be very helpful in the detection of problem areas on a transmission or distribution line and in the removal of persistent faults. For these reasons, fault detection and location has been an enduring preoccupation of power system researchers, designers, and maintenance engineers.

A variety of fault location schemes have been developed over the years. Common systems include impedance-based locators [1,2], or those which measure the impedance seen by one or both ends of the transmission line, and traveling wave-based locators [8-10], or those which rely on the timing of fault detections. In addition to categorizing these fault location methods by the way in which they locate faults, they can also be classified into one-terminal and two-terminal based on whether they require information from one end or both ends of the transmission line, respectively.

2.1. Impedance-Based Methods

Traditional impedance-based fault location methods use the voltages and currents at one or both ends of a transmission line to determine where a fault has occurred. The impedance of the transmission line per unit length is usually required in these calculations. One of the major problems with basic one-terminal impedance-based fault location methods – those that only use measurements from one end of the transmission

line – is that the fault impedance must be near-zero for the result to be accurate, since the fault impedance affects the impedance seen at the end of the transmission line [3]. This problem has been mitigated in several different ways. One of the best-known of these ways is the Takagi method. This changes the calculation to include the difference between the current measured before the fault and the current measured after the fault (which is the fault current) [1]. This eliminates the fault impedance from the analysis, thus removing this significant source of error. However, the angle of the fault current and the angle of the current during the fault at the relay terminals are assumed to be equal; if this is not true, there may be errors in the fault location.

Two-terminal impedance-based fault location methods, or those that use measurements from both ends of the transmission line, can also significantly improve the accuracy of the fault location estimate [4]. Two-terminal methods require communication between the locators at both ends of the transmission line to transfer information about the currents, voltages, and source impedances in order to perform the fault location. Once all of the necessary information is gathered in one location, the fault is located by combining the equations describing both sides of the transmission line and variables directly related to the fault; the exact analysis depends on the particular algorithm [2,3]. The use of both sides of the transmission line in calculation removes most of the problems associated with one-terminal impedance-based fault location methods. It is essential to note, however, that short-duration faults are difficult to accurately detect with any impedance-based fault location methods (although two-terminal methods reduce the effect of short fault durations) since less data is available about the voltages and currents and the data that is available is not necessarily in steady-state [4].

2.2. Traveling Wave-Based Methods

Traveling wave-based fault location methods, like impedance-based methods, can be divided into one-terminal and two-terminal methods. With traveling wave analysis, however, the entire method of location rather than simply the equations change between the one- and two-terminal methods. One-terminal methods rely on the timing between reflections of voltage or current at impedance discontinuities – in this case, the fault – to find the distance between the sensor and the fault while two-terminal methods work based on the time delay between arrivals of information at the ends of the transmission line.

Traveling wave-based fault location methods have been divided into five distinct Types [5,6]. Type A is a one-terminal location method which calculates the fault location based on the time between the first detection of a fault and the detections of reflections of the transient generated by the fault. Type B is a two-terminal method in which as the locator at each end of the transmission line detects a fault, it sends a signal to the other end of the transmission line. The time of the signal's arrival is used in the fault location timing. The method of the signal transmission and the possibility of a delay between the fault detection and the generation of the signal can vary depending on the chosen sub-Type. Type C is a one-terminal method and is much like type A, except it uses a generated pulse and its reflections to locate the fault rather than using the fault transient and its reflections. Type D is a two-terminal method which uses the detection times of the transients at opposite ends of the transmission line to determine the fault location; the locators at both ends of the transmission line must be synchronized for this Type to work properly. Finally, Type E is a one-terminal method which uses the transients produced

when the circuit breaker re-energizes the line in order to locate persistent faults. Types A, D, and sometimes E are the Types which are used most frequently in modern traveling wave-based fault locators [5]; GPS has made Type D locators especially attractive since it provides a method for synchronization of the fault locators at the two ends of the transmission line [7]. Such a system is depicted in Figure 2-1. This enables an accurate measurement of the difference between two fault detection times and thus a more accurate location of the fault.

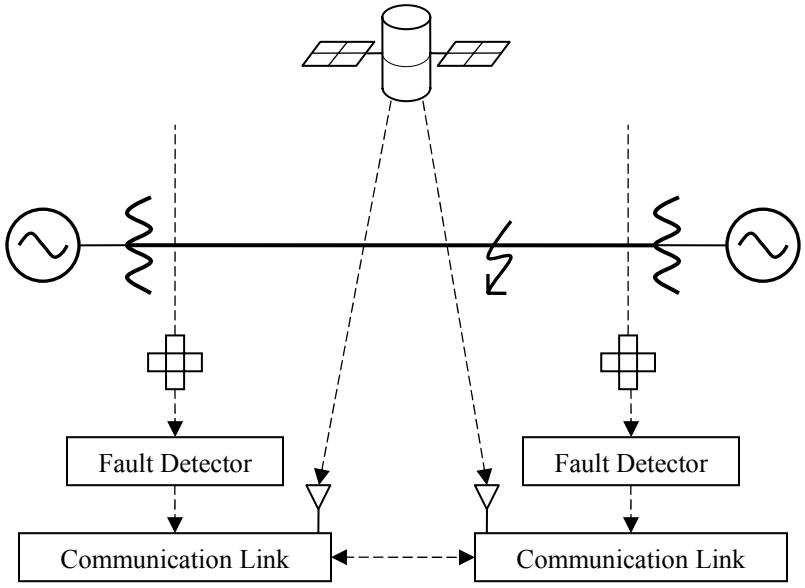


Figure 2-1 – GPS satellite being used to synchronize fault detection timing

Traveling wave-based fault locators, both two-terminal and one-terminal, are still being designed. Many of these make use of the previously defined Types of fault locators in combination with other algorithms such as autocorrelation [8] and the wavelet transform [9]. The wavelet transform is similar to the Fourier transform in that it decomposes a signal into frequency components, but it also localizes these components in time. When it is applied to fault detection and location, the wavelet transform is often used in conjunction with traveling wave-type fault locators as the method of fault and

reflection detection [10]. Faults can be easily detected by monitoring the magnitudes of the individual frequency scales.

The algorithms presented in this thesis are most similar to Type D traveling wave-based location schemes. Traveling wave-based locators can be very accurate, provided the time of fault arrival (and reflection arrival, for the algorithms that make use of reflections) can be detected accurately. Problems can arise, however, with faults that occur at the zero-crossing of the transmission line voltage or current since the resulting change in the waveform is not particularly pronounced. This is a more significant problem for high fault impedances; for comparatively low fault impedances, a well-designed traveling wave-based fault location algorithm will still be able to locate all faults with a great deal of accuracy.

2.3. Detection and Location Using Magnetic Field Sensors

Due to the simple relationship between current and magnetic field intensity, it is understandable that magnetic field sensors have previously been used in fault detection and location schemes. These schemes often use magnetic field sensors in place of current transformers since magnetic field sensors can be installed independently from a substation or switching station with a minimum amount of additional equipment [11,12].

One possible use of this relationship is simply replacing each current transformer with a Hall effect transducer. This transducer would typically need to be within the electrical arcing distance of the conductors to produce enough voltage for analysis and would thus require insulation. To remove this need for insulation, the transducer can be located between two tapered pieces of ferromagnetic material in order to concentrate the magnetic field into the transducer [11]; as a result, the transducer does not need to be

located within the arcing distance of the conductors. The measured magnetic field result can then be used similarly to a current measurement for fault detection and location.

Due to the reduced amount of equipment needed for analysis when compared with current transformers, it is possible to use several sets of magnetic field intensity sensors on a single transmission line. For example, one patent [12] suggests the installation of magnetic field intensity sensors on every pole of a transmission or distribution line as a distributed fault detection and location system. Phase to ground faults are detected using magnetic sensors around the ground conductors; phase to phase faults are detected with a sensor which detects orthogonal fields due to arcing. If a fault occurs, it is most likely between the first two poles at which it is detected, and thus any further searching for the fault only needs to be within that area. This requires only a minimal synchronization between the multiple sensors, since the fault location is not based upon exact time of the fault incidence but simply the first locations at which the fault was detected. This system is conceptually interesting but rather expensive due to the number of sensors and microprocessors that would be required, even if sensors are only located on one out of every few poles.

Since a magnetic field sensor does not need to make contact with the conductors [11] and can be installed remotely from substations [12], the effectiveness of the magnetic field sensor in fault detection and location algorithms is clearly worth examining.

3. General Design of the System

As previously stated, a three-phase system can be analyzed by detecting the vertical and horizontal magnetic field intensities and comparing the waveforms to the expected results. In order to better understand the algorithms which will be used for this analysis, a basic mathematical and physical description of the magnetic field near a transmission line is presented here.

3.1. Magnetic Field Analysis of a Generalized System

The magnetic field near a transmission or distribution line can be determined for the general case of n conductors. The necessary analysis follows the work by Emanuel, Orr, Pileggi, and Gulachenski [13], although some modifications have been made in this summary of their work to accommodate differences between the magnitudes of the conductor currents. The system to be analyzed is shown in Figure 3-1.

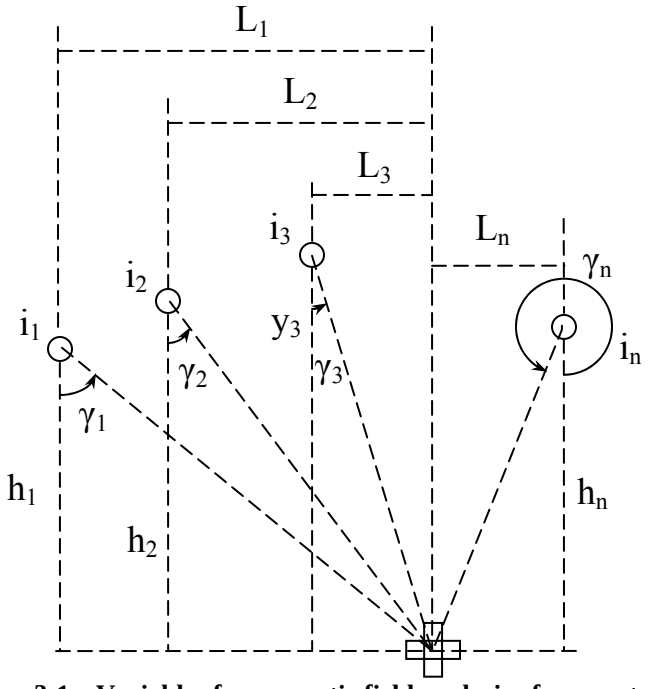


Figure 3-1 – Variables for magnetic field analysis of any system

The magnetic fields due to any given conductor k are

$$H_{x,k} = D_k i_k \text{ and } H_{y,k} = Q_k i_k \tag{1}$$

where

$$D_k = \frac{\cos^2 \gamma_k}{2\pi h_k} \text{ and } Q_k = \frac{\sin \gamma_k \cos \gamma_k}{2\pi h_k} \tag{2}$$

and

$$\gamma_k = \tan^{-1}\left(\frac{L_k}{h_k}\right) \tag{3}$$

Each current can be described as a sum of a sine and cosine, such that

$$i_k = \sqrt{2}\left[X_k \sin(\omega t) + Y_k \cos(\omega t)\right] \tag{4}$$

where X_k and Y_k are real numbers. Since

$$X_k \sin(\omega t) + Y_k \cos(\omega t) = I_k \sin(\omega t + \varphi_k) \tag{5}$$

where φ_k is the phase shift of the current and I_k is the RMS value of the current, and

$$\varphi_k = \tan^{-1}\left(\frac{Y_k}{X_k}\right) \tag{6}$$

the terms X_k and Y_k can be described as

$$X_k = \frac{I_k}{\sqrt{1 + \tan^2(\varphi_k)}} \tag{7}$$

and

$$Y_k = \frac{I_k \tan(\varphi_k)}{\sqrt{1 + \tan^2(\varphi_k)}} \tag{8}$$

Using the trigonometric identity

$$1 + \tan^2(\varphi_k) = \sec^2(\varphi_k) \tag{9}$$

the values of X_k and Y_k can be simplified to

$$X_k = I_k \cos(\varphi_k) \tag{10}$$

and

$$Y_k = I_k \sin(\varphi_k) \tag{11}$$

The currents and their coefficients can then be written as

$$[i] = \begin{bmatrix} i_1 \\ i_2 \\ i_3 \\ \vdots \\ i_n \end{bmatrix} \text{ and } [XY] = \begin{bmatrix} I_1 & 0 \\ X_2 & Y_2 \\ X_3 & Y_3 \\ \vdots & \vdots \\ X_n & Y_n \end{bmatrix} \tag{12}$$

The magnetic field due to all of the currents is then given by

$$[H] = [P][i] \tag{13}$$

where the magnetic field matrix is

$$[H] = \begin{bmatrix} H_x \\ H_y \end{bmatrix} \tag{14}$$

and the position matrix is

$$[P] = \begin{bmatrix} D_1 & D_2 & D_3 & \dots & D_n \\ Q_1 & Q_2 & Q_3 & \dots & Q_n \end{bmatrix} \tag{15}$$

The magnetic field can thus be described by

$$H_x = \sqrt{2}\left[A_x \sin(\omega t) + B_x \cos(\omega t)\right] = \hat{H}_x \sin(\omega t + \phi_x) \tag{16}$$

and

$$H_y = \sqrt{2}\left[A_y \sin(\omega t) + B_y \cos(\omega t)\right] = \hat{H}_y \sin(\omega t + \phi_y) \tag{17}$$

where

$$\hat{H}_x = \sqrt{2\left(A_x^2 + B_x^2\right)} \text{ and } \phi_x = \tan^{-1}\left(\frac{B_x}{A_x}\right) \tag{18}$$

and

$$\hat{H}_y = \sqrt{2\left(A_y^2 + B_y^2\right)} \text{ and } \phi_y = \tan^{-1}\left(\frac{B_y}{A_y}\right) \tag{19}$$

The coefficients A_x, A_y, B_x and B_y can be found using the relationship

$$\begin{bmatrix} A_x & B_x \\ A_y & B_y \end{bmatrix} = [P][XY] \tag{20}$$

In order to determine the shape of the magnetic field, Equation (16) can be solved for ωt, which results in

$$\omega t = \sin^{-1}\left(\frac{H_x}{\hat{H}_x}\right) - \phi_x \tag{21}$$

This in turn can be substituted into Equation (17), so that

$$H_y = \hat{H}_y \sin\left(\sin^{-1}\left(\frac{H_x}{\hat{H}_x}\right) + \phi_y - \phi_x\right) \tag{22}$$

Substituting the variable

$$\phi = \phi_y - \phi_x \tag{23}$$

this can be rewritten as

$$H_y = \hat{H}_y\left[\sin\left(\sin^{-1}\left(\frac{H_x}{\hat{H}_x}\right)\right)\cos(\phi) + \cos\left(\sin^{-1}\left(\frac{H_x}{\hat{H}_x}\right)\right)\sin(\phi)\right] \tag{24}$$

which results in

$$H_y = \frac{\hat{H}_y}{\hat{H}_x} H_x \cos(\phi) + \frac{\hat{H}_y}{\hat{H}_x}\sqrt{\hat{H}_x^2 - H_x^2}\,\sin(\phi) \tag{25}$$

since

$$\cos\left(\sin^{-1}\left(\frac{H_x}{\hat{H}_x}\right)\right)=\frac{\sqrt{\hat{H}_x^2-H_x^2}}{\hat{H}_x} \tag{26}$$

This can be simplified further by defining

$$r=\frac{\hat{H}_y}{\hat{H}_x} \tag{27}$$

which gives

$$H_y-rH_x\cos(\phi)=r\sqrt{\hat{H}_x^2-H_x^2}\sin(\phi) \tag{28}$$

After squaring both sides of the equation and canceling terms, this simplifies to

$$H_y^2+r^2H_x^2-2rH_xH_y\cos(\phi)=\hat{H}_y^2\sin(\phi) \tag{29}$$

This is the equation for an ellipse with the angle of rotation equal to

$$\alpha=\frac{1}{2}\tan^{-1}\left(\frac{2r\cos(\phi)}{1-r^2}\right) \tag{30}$$

and half axes of lengths

$$\hat{H}_a=\frac{\hat{H}_y\sin(\phi)}{\sqrt{\sin^2\alpha+r^2\cos^2\alpha+r\cos(\phi)\sin(2\alpha)}} \tag{31}$$

and

$$\hat{H}_b=\frac{\hat{H}_y\sin(\phi)}{\sqrt{\cos^2\alpha+r^2\sin^2\alpha+r\cos(\phi)\sin(2\alpha)}} \tag{32}$$

An example elliptical rotating magnetic field which could be produced by the combined magnetic fields of several conductors is presented in Figure 3-2. This is also shown as a part of Figure 3-7. All Figures which present magnetic fields in polar coordinates display rho in units of amperes per meter and theta in units of degrees.

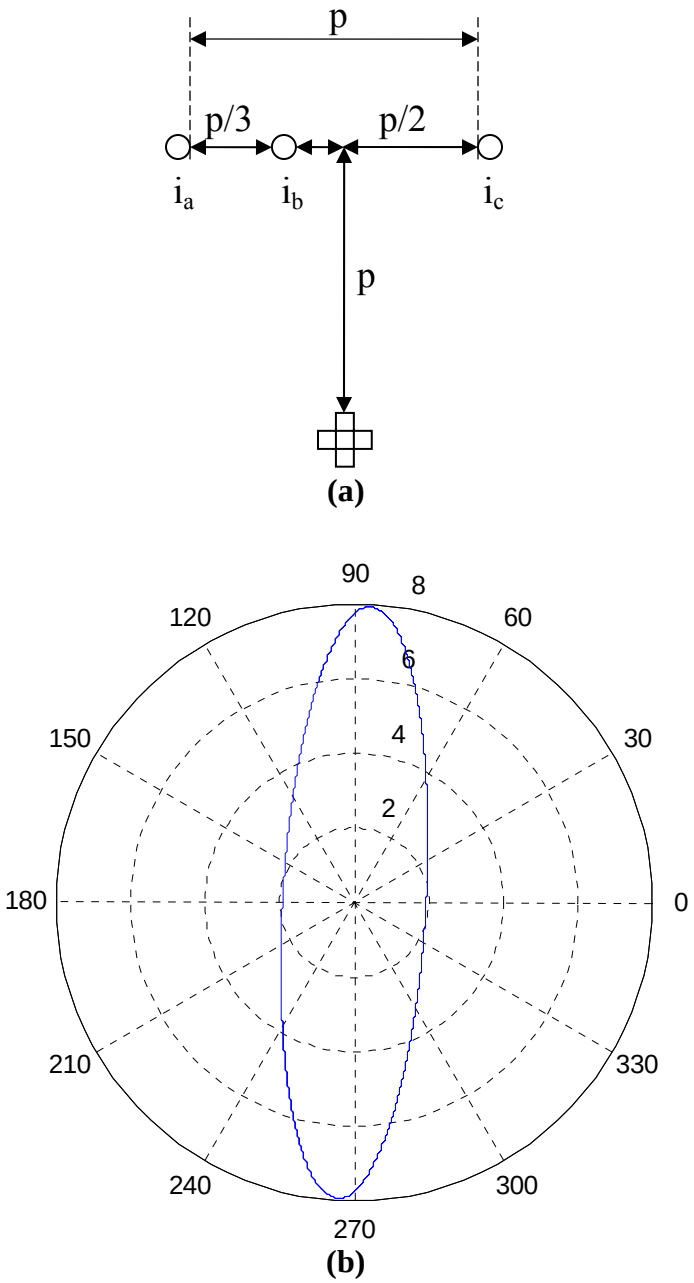


Figure 3-2 – Elliptical rotating magnetic field produced by three-phase, three-conductor system (a) – Conductor geometry; (b) – Magnetic field for p=2m

This analysis is essential to understanding the magnetic field which will be detected by the sensors. It is applicable to any configuration of conductors, but it is further analyzed here for a three-conductor system.

3.2. Magnetic Field Analysis of a Three-Conductor System

The three-conductor system shown below in Figure 3-3 is used in the simulation of the magnetic field intensity detected by the sensors. The variables used make this analysis something of a general case which can be modified for the analysis of most transmission line structures. The assumption is made, however, that the sensors are located directly below phase b.

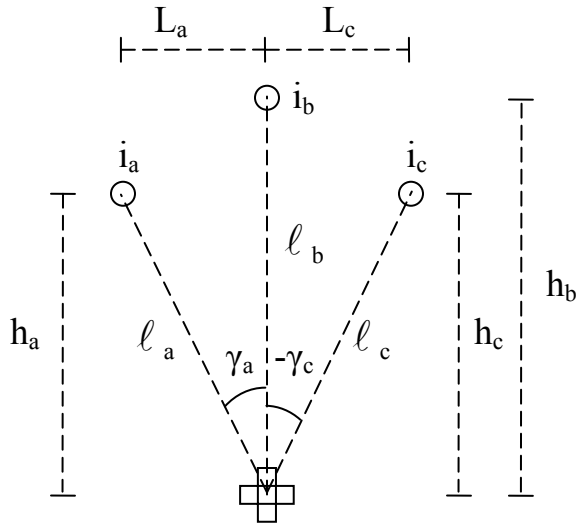


Figure 3-3 – Variables for magnetic field analysis of a three-phase system

The distances ℓ_a, ℓ_b , and ℓ_c are can be related to the other distances by the equations

$$\ell_a = \sqrt{L_a^2 + h_a^2} \tag{33}$$

$$\ell_b = h_b \tag{34}$$

$$\ell_c = \sqrt{L_c^2 + h_c^2} \tag{35}$$

These equations are simplified (and the following analysis is simplified) when $L_a = L_c$ and $h_a = h_c$ (and thus $\ell_a = \ell_c$), as would be the case for many transmission line configurations, but at this point in the analysis this assumption is ignored.

The component identified as the magnetic sensor is comprised of two separate magnetic sensors which will independently detect the horizontal and vertical magnetic field intensities. These field intensities are determined by the sums of the magnetic field intensities due to each conductor, which are given by

$$H_a = \frac{i_a}{2\pi\ell_a} \tag{36}$$

$$H_b = \frac{i_b}{2\pi\ell_b} \tag{37}$$

$$H_c = \frac{i_c}{2\pi\ell_c} \tag{38}$$

where H_a, H_b, and H_c and i_a,i_b, and i_c are the magnetic field intensities and currents, respectively. These can be converted into the horizontal and magnetic field intensities using the equations

$$H_x = H_a \cos(\gamma_a) + H_b + H_c \cos(-\gamma_c) \tag{39}$$

$$H_y = H_a \sin(\gamma_a) + H_c \sin(-\gamma_c) \tag{40}$$

where

$$\cos(\gamma_a) = h_a / \ell_a \tag{41}$$

$$\cos(\gamma_c) = h_c / \ell_c \tag{42}$$

$$\sin(\gamma_a) = L_a / \ell_a \tag{43}$$

$$\sin(\gamma_c) = -L_c / \ell_c \tag{44}$$

Substituting these results,

$$H_x = i_a\left(h_a / \ell_a^2\right) + i_b / \ell_b + i_c\left(h_c / \ell_c^2\right) \tag{45}$$

$$H_y = i_a\left(L_a/\ell_a^2\right) - i_c\left(L_c/\ell_c^2\right) \tag{46}$$

As previously stated, these can be simplified further if phases a and c are at the same vertical and horizontal distances from the center point.

This can also be rewritten by expressing the currents as functions of time, as

$$H_x = |i_a|\sin(\omega t+\varphi_a)\left(h_a/\ell_a^2\right) + \left(|i_b|\sin(\omega t+\varphi_b)\right)/\ell_b + |i_c|\sin(\omega t+\varphi_c)\left(h_c/\ell_c^2\right) \tag{47}$$

$$H_y = |i_a|\sin(\omega t+\varphi_a)\left(L_a/\ell_a^2\right) - |i_c|\sin(\omega t+\varphi_c)\left(L_c/\ell_c^2\right) \tag{48}$$

Additionally, restating parts of Equations (16) and (17), these equations can be written as

$$H_x = \sqrt{2}\left[A_x\sin(\omega t) + B_x\cos(\omega t)\right]$$

and

$$H_y = \sqrt{2}\left[A_y\sin(\omega t) + B_y\cos(\omega t)\right]$$

Following the analysis in Section 3.1 and specifically Equations (10), (11) and (20), the coefficients A_x, A_y, B_x, and B_y can be found using the equation

$$\begin{bmatrix} A_x & B_x \\ A_y & B_y \end{bmatrix} = \begin{bmatrix} \dfrac{h_a}{2\pi\left(h_a^2+L_a^2\right)} & \dfrac{1}{2\pi h_b} & \dfrac{h_c}{2\pi\left(h_c^2+L_c^2\right)} \\ \dfrac{L_a}{2\pi\left(h_a^2+L_a^2\right)} & 0 & \dfrac{-L_c}{2\pi\left(h_c^2+L_c^2\right)} \end{bmatrix} \begin{bmatrix} I_a\cos(\varphi_a) & I_a\sin(\varphi_a) \\ I_b\cos(\varphi_b) & I_b\sin(\varphi_b) \\ I_c\cos(\varphi_c) & I_c\sin(\varphi_c) \end{bmatrix} \tag{49}$$

which simplifies to

$$\begin{bmatrix} A_x & B_x \\ A_y & B_y \end{bmatrix} = \begin{bmatrix} \dfrac{h_a}{2\pi\left(h_a^2+L_a^2\right)} & \dfrac{1}{2\pi h_b} & \dfrac{h_c}{2\pi\left(h_c^2+L_c^2\right)} \\ \dfrac{L_a}{2\pi\left(h_a^2+L_a^2\right)} & 0 & \dfrac{-L_c}{2\pi\left(h_c^2+L_c^2\right)} \end{bmatrix} \begin{bmatrix} I_a & 0 \\ I_b\cos(\varphi_b) & I_b\sin(\varphi_b) \\ I_c\cos(\varphi_c) & I_c\sin(\varphi_c) \end{bmatrix} \tag{50}$$

if the phase shift in phase a is arbitrarily set as $\varphi_a = 0$.

The ground conductors also can contribute to these magnetic fields when a fault occurs, but this contribution will be minimal unless the fault is very close to the location

of the detector. This is because most of the ground currents will flow directly into the earth prior to reaching the substation or switching station and thus the ground wires will typically only contribute a negligible amount to the magnetic field at the substation or switching station, even for a significant fault current [14]. As a result, the ground conductors have not been included in this analysis.

3.3. Conductor Configurations and the Magnetic Field

The magnetic field which is detected by the current sensors will change based on the configuration of the conductors. Since the magnetic field-based fault detector should be able to be used in a variety of situations, it is essential to gain an understanding of its performance for several conductor configurations. The magnetic fields of five different conductor configurations are demonstrated below in Figure 3-4 through Figure 3-8. Each Figure shows the conductor configuration, the magnetic field during balanced operation, the magnetic field during a single line to ground fault on phase a, and the magnetic field during a line to line fault on phases a and b. The currents in the unfaulted conductors have an RMS value of 100A; the fault currents are approximately 1000A RMS. The value of p was set as 2 meters, and the conductors are assumed to correspond to phases a, b, and c from left to right.

3.3.1. Horizontal Conductor Configuration

Figure 3-4 shows a horizontal conductor configuration and its magnetic fields. The sensors are located a distance p below phase b where p is the distance between phases a and c. As expected, the vertical component of the magnetic field during normal operation is greater than the horizontal component. The vertical maximums occur when

the current in phase b is crossing zero and as a result, the other phases cancel each others' horizontal components but add in the same direction vertically. Likewise, the horizontal maximums occur when the current in phase b is at a maximum; however, at this time the horizontal components of the magnetic fields due to phases a and c add in the opposite direction and cancel some of the horizontal magnetic field due to phase b, while the vertical components due to phases a and c cancel each other.

The magnetic field of the single line to ground fault on phase a is as expected. The magnetic field contributions due to phases b and c become relatively insignificant, and the magnetic field is essentially only the field due to phase a. The angle of rotation is also mostly determined by the angle between the phase a conductor and the sensors from a vertical reference. Neglecting the effects of the other phases, the angle of rotation would be

$$\tan^{-1}\left(\frac{p/2}{p}\right) = \tan^{-1}(0.5) \approx 26.6°$$

which is fairly close to the observed angle of rotation. The difference is due to the fact that this calculated angle does not account for the currents in phases b and c.

The line to line fault on phases a and b creates a magnetic field which is somewhat unexpected, particularly with respect to the angle of rotation. A brief description will clarify this matter, however. Since the magnitudes of the currents in phases a and b are identical in this situation and they are approximately 180° out of phase due to the line to line fault, the positive maximum horizontal field due to phase a (which occurs when the current is at its positive maximum) is counteracted by a field inclined at the same angle as the angle between the phase a conductor and the sensors, as noted above. As a result, this maximum magnetic field has a positive horizontal component and

a negative vertical component. The same result in the opposite direction occurs when the current in phase a is at its negative maximum and phase b is at its positive maximum. It is clear from these plots that, at least for a significant fault current, a line to ground or line to line fault could be detected using magnetic sensors for this conductor configuration.

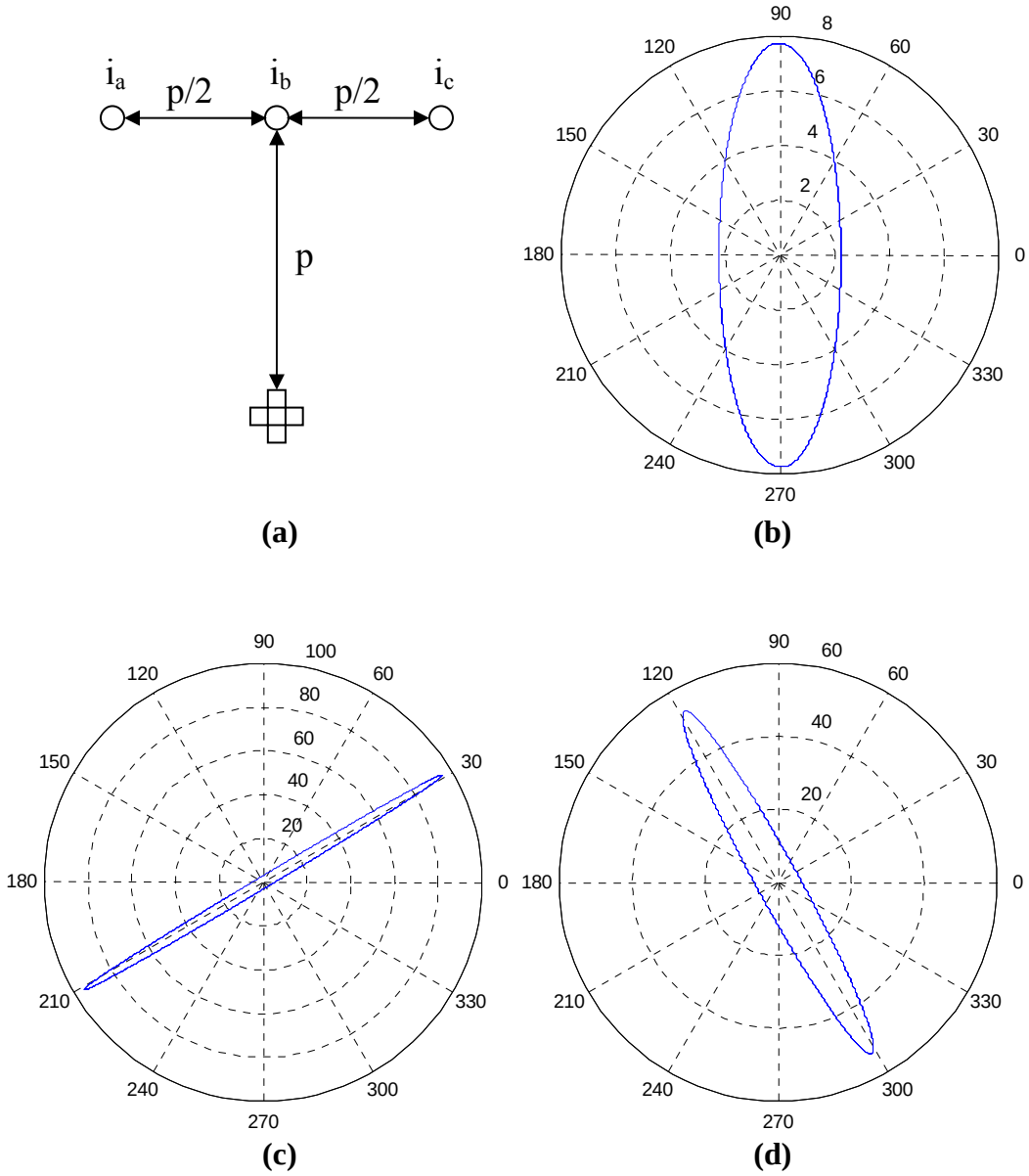


Figure 3-4 – Horizontal conductor configuration and plots of its magnetic field
(a) – Conductor geometry; (b) – Magnetic field for balanced currents, p=2m;
(c) – Magnetic field for a line to ground fault on phase a, p=2m; (d) – Magnetic field for line to line fault on phases a and b, p=2m

3.3.2. Delta Conductor Configuration

The conductor configuration shown in Figure 3-5 is a delta configuration. The magnetic fields here are similar to those of the horizontal conductor configuration under normal operating conditions and for the single line to ground fault on phase a. However, the line to line fault on phases a and b has a very different angle of rotation. The reason for this is that since the magnetic field is inversely proportional to the square of the distance from the conductor, the contribution from phase b is significantly reduced compared to its contribution for the case of the horizontal conductor configuration. As a result, while the analysis above regarding the horizontal configuration remains true, the magnitude of the maximum magnetic field due to phase b is reduced. Thus, the magnetic field is more significantly affected by the current in phase a – significantly enough that the maximum positive horizontal component due to phase b is less than the negative horizontal component due to phase a at the same time. This in turn causes the significant change in the angle of rotation of the ellipse.

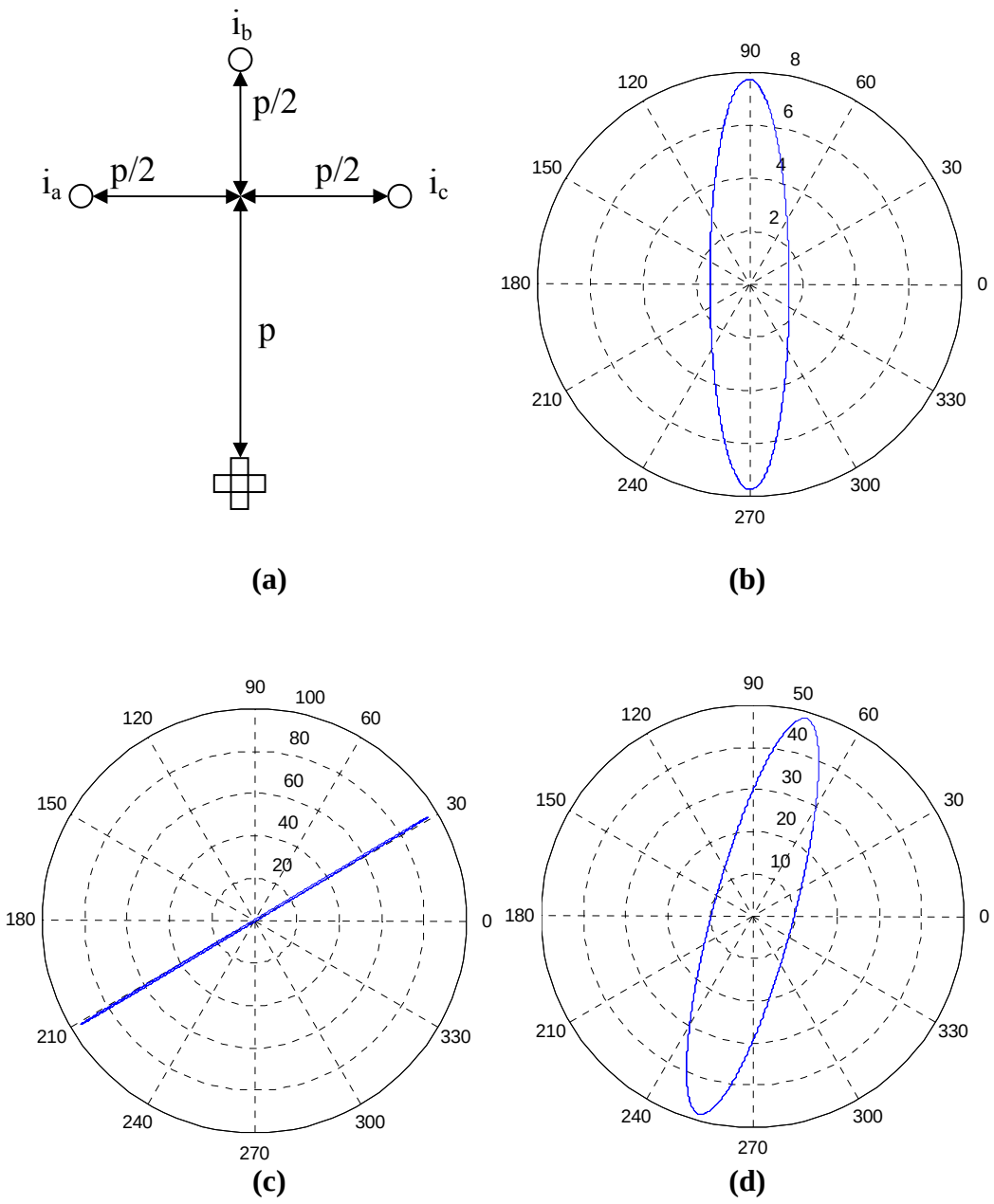


Figure 3-5 – A delta conductor configuration and plots of its magnetic field
(a) – Conductor geometry; (b) – Magnetic field for balanced currents, p=2m;
(c) – Magnetic field for a line to ground fault on phase a, p=2m; (d) – Magnetic field for line to line fault on phases a and b, p=2m

3.3.3. Another Delta Conductor Configuration

A different delta conductor configuration and its magnetic fields are presented in Figure 3-6. These results are similar to those of the previously described delta

configuration and the changes are as expected due to the previous changes in results between the horizontal conductor configuration and the first delta configuration. The increase in the horizontal maximum of the total magnetic field is interesting since it decreased previously. This is since for the horizontal configuration, the horizontal magnetic field maximums are dominated by the current in phase b. For the first delta configuration, this contribution has decreased. As phase b is further removed from the sensors, the other two conductors begin to take over the maximum horizontal magnetic field. The vertical field magnitudes have not changed since phase b does not contribute to the vertical magnetic field and phases a and c have not been moved for these configurations.

The magnetic field plot of the single line to ground fault on phase a has not changed noticeably, as expected. The magnetic field for the line to line fault on phases a and b is more significantly rotated than before, since phase b has been moved further away from the sensors.

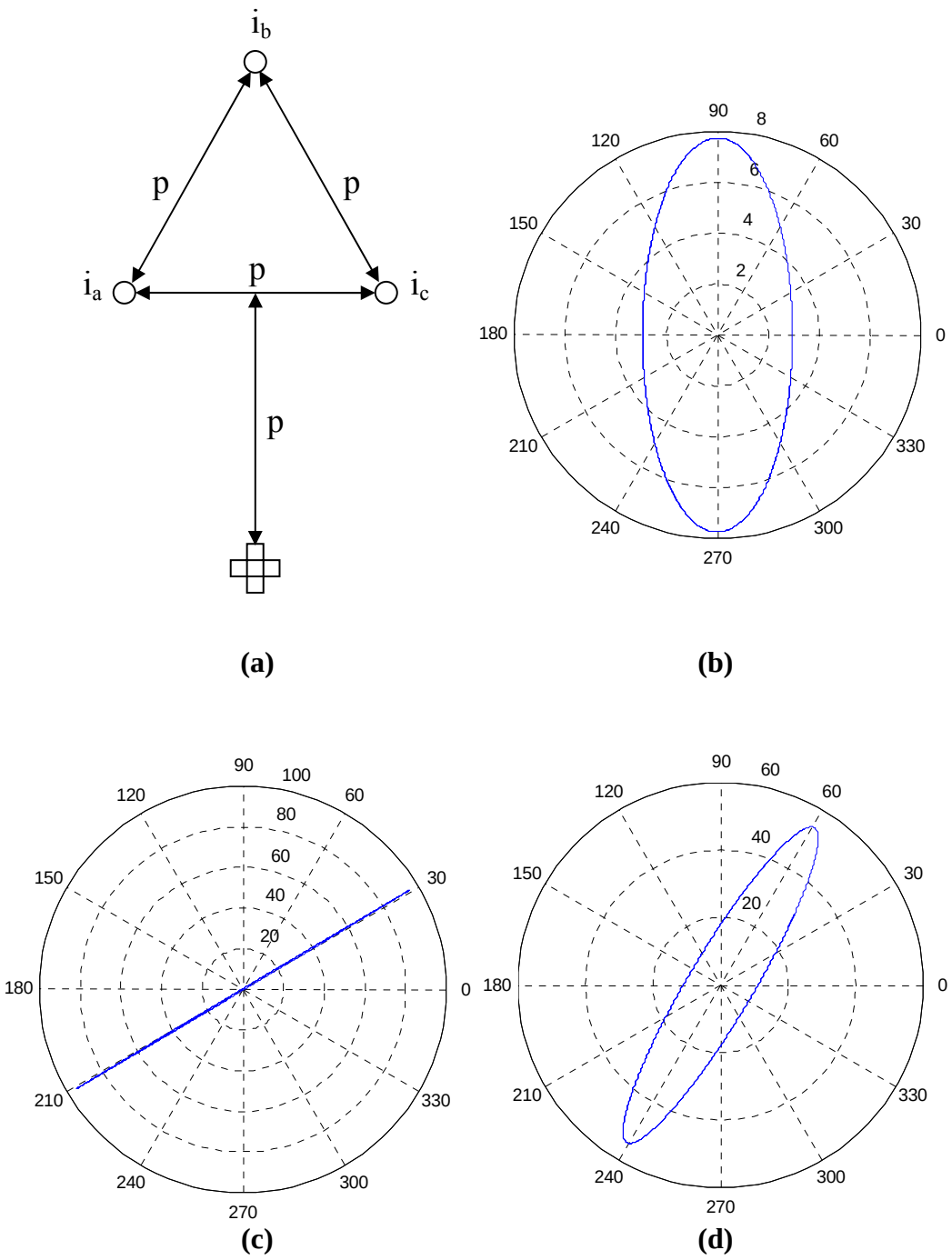


Figure 3-6 – A different delta conductor configuration and plots of its magnetic field
(a) – Conductor geometry; (b) – Magnetic field for balanced currents, p=2m;
(c) – Magnetic field for a line to ground fault on phase a, p=2m; (d) – Magnetic field for line to line fault on phases a and b, p=2m

3.3.4. Horizontal Configuration with Uneven Conductor Spacing

It is not uncommon for conductors to be unevenly spaced, especially in situations where a single ground wire is strung with the transmission line. This type of configuration is shown in Figure 3-7. If the magnetic field sensors are located beneath the point halfway between the conductors for phases a and c, the magnetic fields will be slightly different than those detected for a typical horizontal conductor configuration such as the configuration in Section 3.3.1. The contributions of phases a and c to the magnetic field remain the same. However, phase b now contributes a bit to the vertical field and a bit less to the horizontal field. As a result, the magnetic field during normal operating conditions is slightly rotated in the negative direction. The single line to ground fault on phase a is not significantly affected, and the line to line fault is rotated from the field in Figure 3-4 in the positive direction in accordance with the description in Section 3.3.1, since the horizontal contribution due to phase b is reduced.

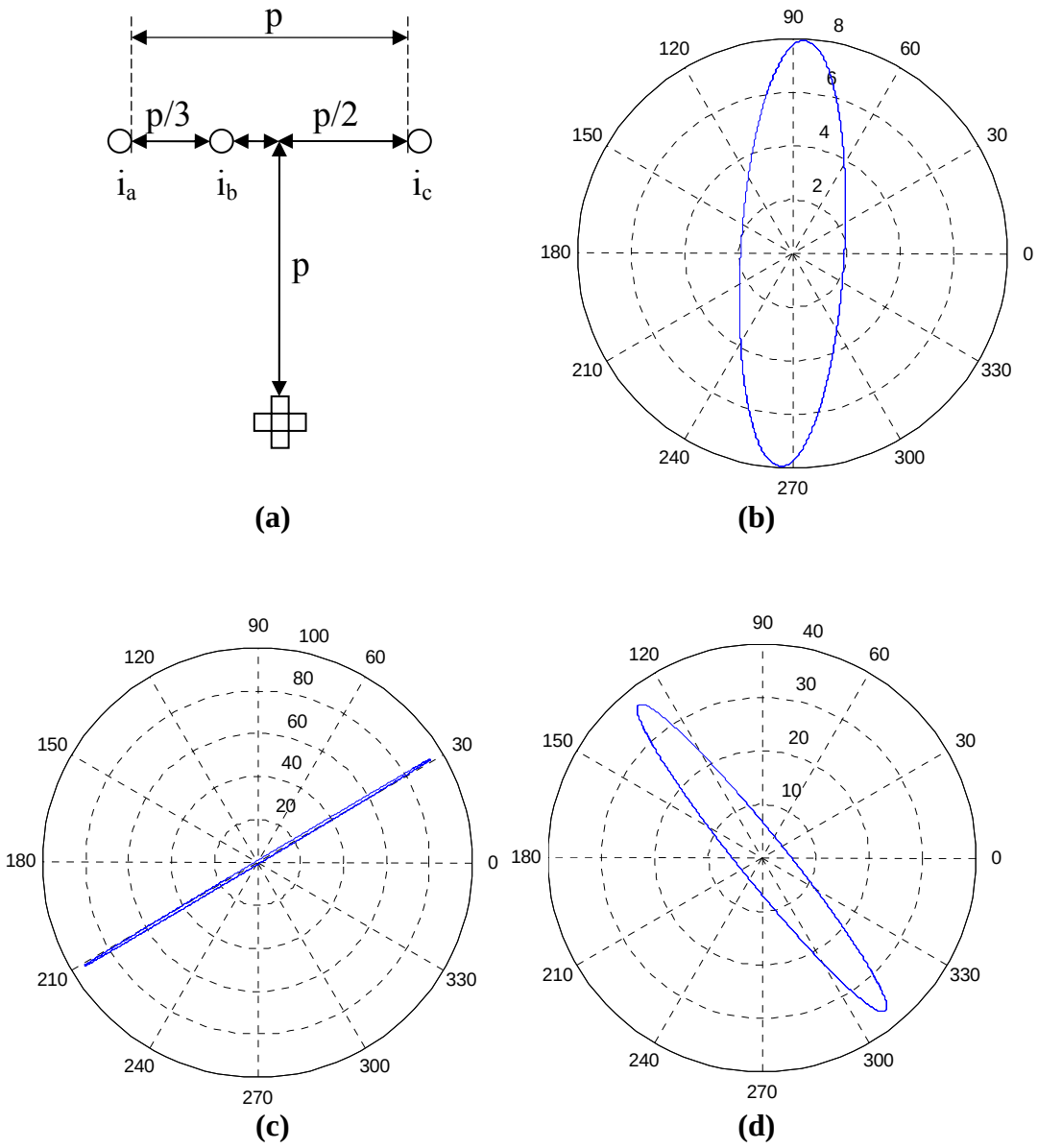


Figure 3-7 – Horizontal configuration with conductors unevenly spaced and plots of its magnetic field (a) – Conductor geometry; (b) – Magnetic field for balanced currents, p=2m; (c) – Magnetic field for a line to ground fault on phase a, p=2m; (d) – Magnetic field for line to line fault on phases a and b, p=2m

3.3.5. Vertical Conductor Configuration

A vertical conductor configuration and the magnetic fields resulting from it are shown in Figure 3-8. It is clear that no vertical magnetic field will be produced from any of the conductors here; the magnitude of the magnetic field changes with the faults, but

higher impedance faults will obviously be much more difficult to detect for this conductor configuration. Incidentally, if the sensors are located a distance p horizontally away from the conductors at the same height as phase b, the resulting magnetic fields will be the same as those in Figure 3-4 except rotated. As a result, the magnetic field sensors can be used for a vertical conductor configuration as well, but they must be mounted to the side of the conductors rather than beneath them.

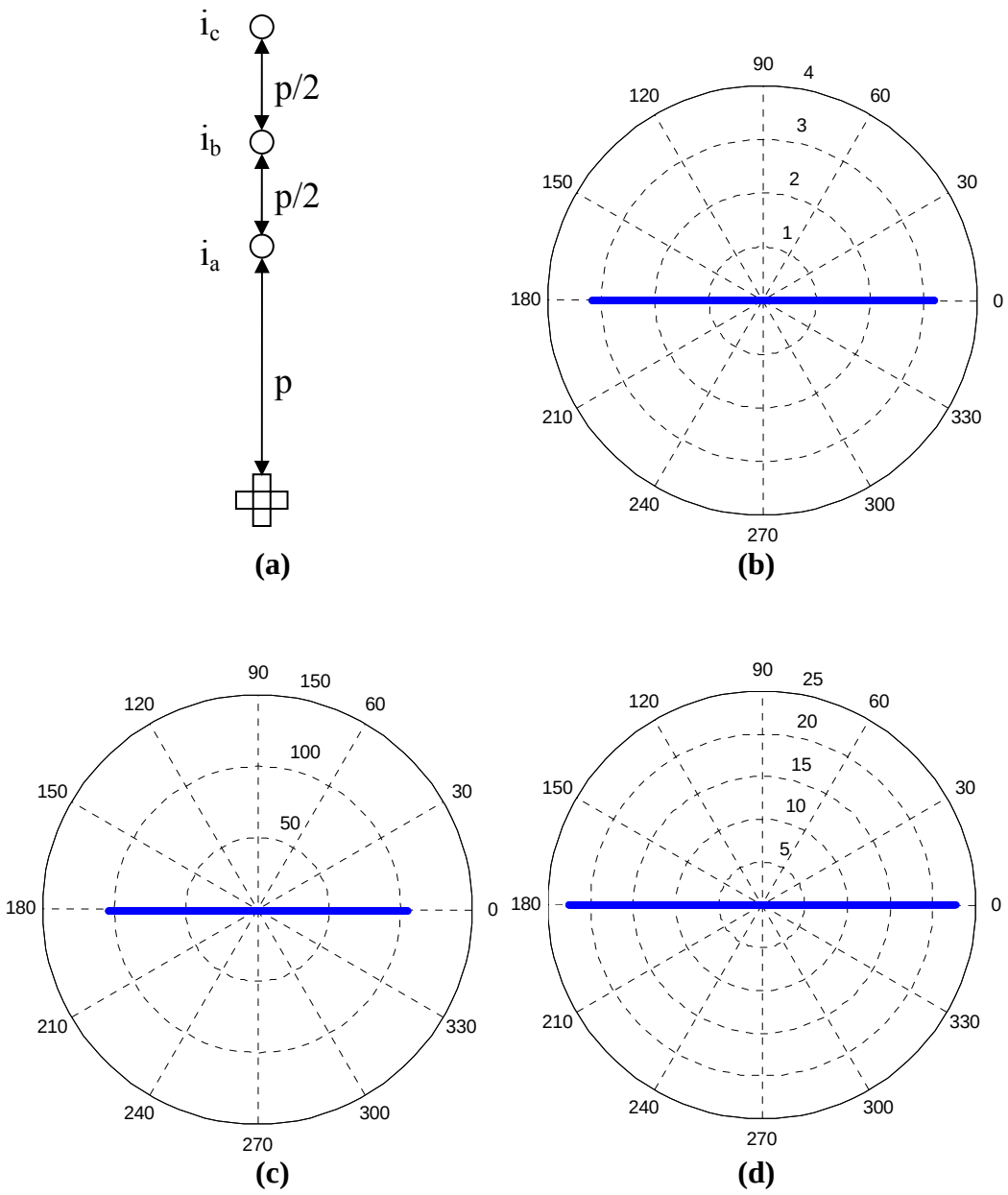


Figure 3-8 – Vertical conductor configuration and plots of its magnetic field
(a) – Conductor geometry; (b) – Magnetic field for balanced currents, p=2m;
(c) – Magnetic field for a line to ground fault on phase a, p=2m; (d) – Magnetic field for line to line fault on phases a and b, p=2m

3.4. Location and Basic Design of the Sensors

By performing this analysis in polar coordinates rather than Cartesian coordinates, the fault detection is based on two variables – rho and theta – but in essence only one of

these is used per algorithm. This makes fault detection more intuitive since only one time-dependent variable needs to be analyzed for each algorithm rather than the three (or more) variables in a typical fault detection and location scheme. The values of rho and theta are given by

$$\rho = \sqrt{H_x^2 + H_y^2} \tag{51}$$

$$\theta = \tan^{-1}\left(\frac{H_y}{H_x}\right) \tag{52}$$

where the result of the inverse tangent has been corrected if H_x is negative.

Assuming a triangular or horizontally coplanar arrangement of conductors, the sensors should ideally be located fairly close to the transmission lines vertically to reduce the effects of magnetic interference since magnetic interference can affect the accuracy of fault detection and, in extreme cases, indicate faults where they do not occur. Also, keeping the sensors fairly close to the transmission lines helps keep the shape of the magnetic field intensity as close to a circle as possible. The shape of the magnetic field intensity ellipse can also cause problems in detection since long, narrow ellipses increase the likelihood of incorrect fault detection. While the magnetic field ellipse could be kept close to a circle in shape in the processing algorithm, it is much more difficult to reduce or eliminate magnetic interference. It should be noted that, although close proximity to the conductors improves the performance of the analysis as stated above, the sensors must be kept outside the arcing distance of any of the conductors.

Additionally, the sensors should be horizontally located directly under the center phase (if possible) to create an elliptical rotating magnetic field that is as close to a circle

as possible and has an angle of rotation that is very close to 0° or 90°. This is again for the purpose of reducing incorrect fault detection.

In order to determine the effects of the horizontal location of the sensors, the magnetic fields were examined for several sensor locations beneath a transmission line. A delta conductor configuration was used for this analysis, and the RMS current in each phase was set at approximately 100A. The value of p in these tests was 2 meters. The results are shown in Figure 3-9 – Figure 3-12.

The basic configuration with the sensors located directly underneath it was already discussed in Section 3.3.2. While this does not merit much more discussion, it is essential to note at this point that the angle of rotation for this conductor location is 90°. The location of the sensors relative to the conductors and the resulting magnetic field are shown below in Figure 3-9.

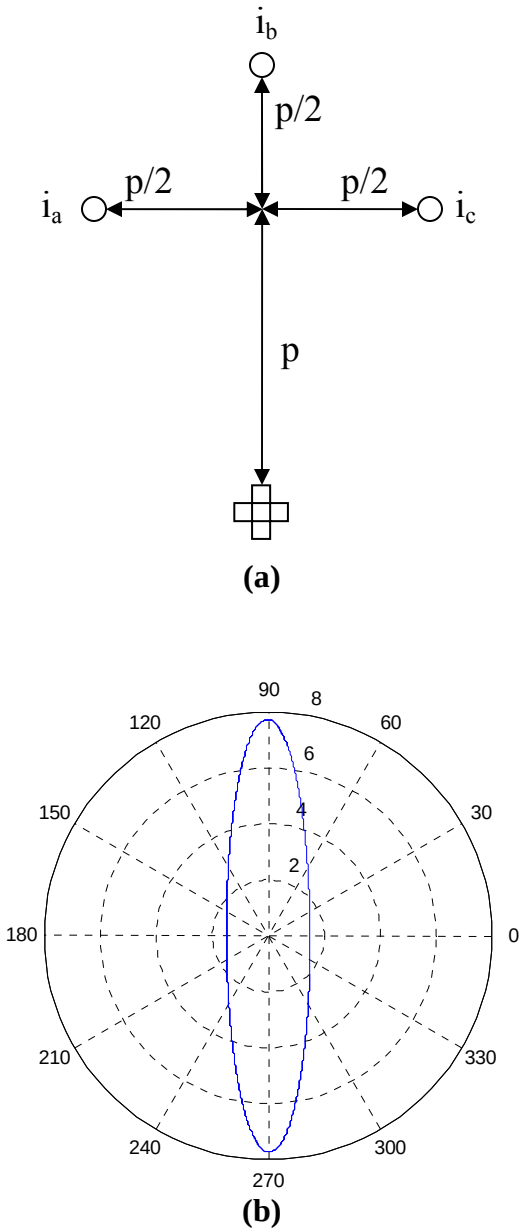
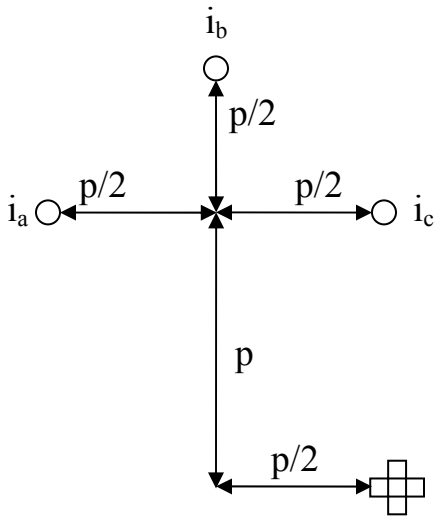


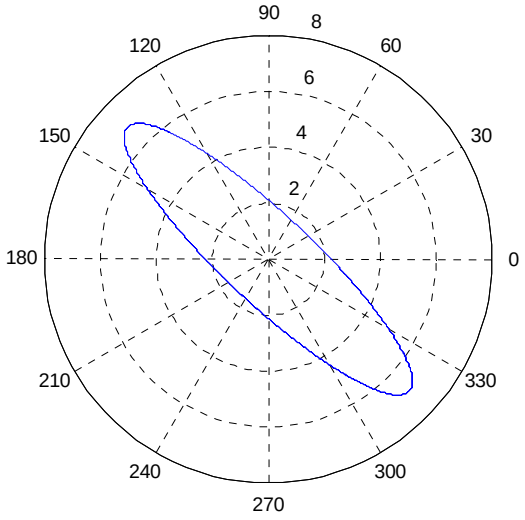
Figure 3-9 – Delta conductor configuration and magnetic field with sensors under center phase (a) – Conductor geometry; (b) – Magnetic field for balanced currents, p=2m

As the sensors are shifted horizontally, the angle of rotation of the elliptical rotating magnetic field begins to increase. The rotation is in this direction because when the horizontal magnetic field due to phase c is at its positive maximum, the currents in phases a and b contribute to the magnetic field negatively, both in the horizontal and vertical directions. This negative contribution in the horizontal direction is not enough to

overcome the contribution due to the current in phase c, so this maximum value of the magnetic field is in the fourth quadrant. A similar situation occurs when the current in phase c is at its negative maximum. A representative sensor location and the resulting magnetic field are shown in Figure 3-10.



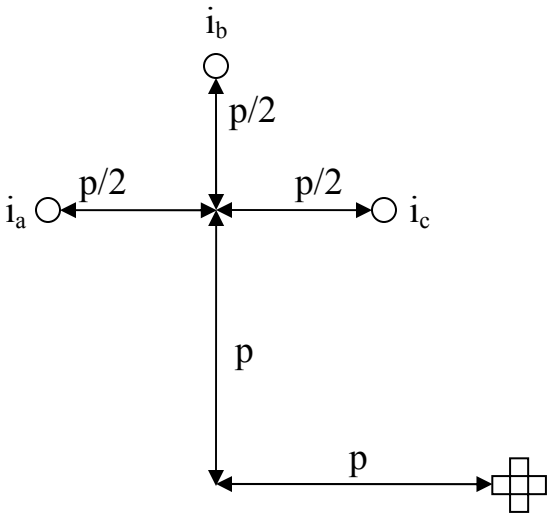
(a)



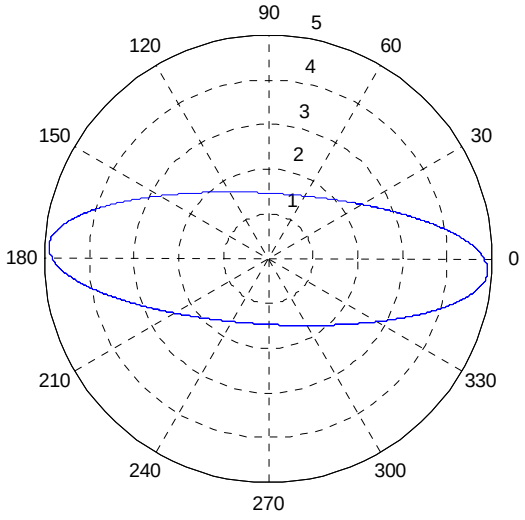
(b)

Figure 3-10 – Delta conductor configuration and magnetic field with sensors shifted
(a) – Conductor geometry; (b) – Magnetic field for balanced currents, p=2m

As the sensors are shifted further, the angle of rotation of the magnetic field continues to increase. The magnitudes of the maximums also begin to decrease; the minimums also decrease, but not at as rapid of a rate. This is simply due to the increased distance from all of the conductors. An example sensor position and the magnetic field at this position are shown in Figure 3-11.



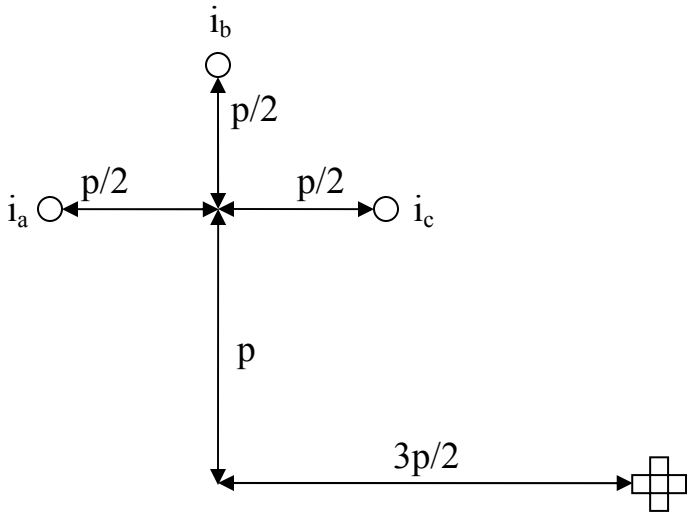
(a)



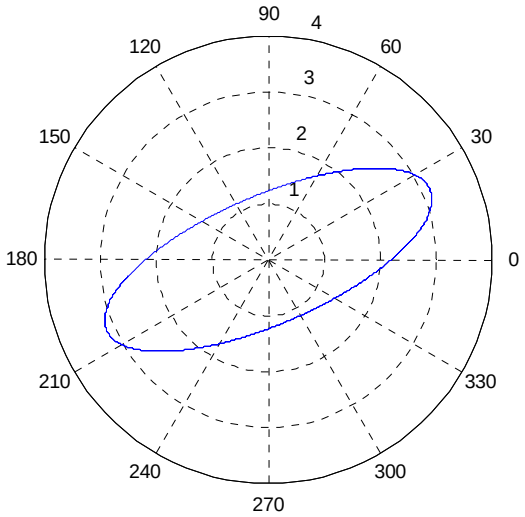
(b)

Figure 3-11 – Delta conductor configuration and magnetic field with sensors shifted farther (a) – Conductor geometry; (b) – Magnetic field for balanced currents, p=2m

As the sensors are shifted even further, the angle of rotation of the elliptical rotating magnetic field increases beyond 180°. Simultaneously, the maximum value of the magnetic field decreases even more. This rotation is due to the increased percentage of the magnetic field resulting from phase c as well as the fact that the angles that all three phases make with the sensors from vertical are becoming closer in value. A sensor location which demonstrates this and the corresponding magnetic field are shown in Figure 3-12.



(a)



(b)

Figure 3-12 – Delta conductor configuration and magnetic field with sensors shifted significantly (a) – Conductor geometry; (b) – Magnetic field for balanced currents, p=2m

In addition to being located under the center of the transmission line and as close to the conductors as possible, the magnetic field sensors should also be located away from magnetic materials or materials that can be magnetized, since these will affect the magnetic field perceived by the sensors. This means that the sensors needs to be mounted

on non-magnetic poles; this is of special note in the case that the pylons used in the transmission system in question are made of material that can be magnetized such as steel.

The suggested designs for the magnetic field sensors are two search coils, one oriented such that its induced voltage is proportional to the vertical magnetic field and one such that its inducted voltage is proportional to the horizontal magnetic field. The voltages induced in the coils are

$$v_x = \mu_0 NA \frac{\partial H_x}{\partial t} = K\omega \hat{H}_x \cos\left(\omega t + \theta_x\right) \tag{53}$$

and

$$v_y = \mu_0 NA \frac{\partial H_y}{\partial t} = K\omega \hat{H}_y \cos\left(\omega t + \theta_y\right) \tag{54}$$

where the coil constant is

$$K = \mu_0 NA \tag{55}$$

and $\mu_0 = 4\pi \times 10^{-7}$ H/m is the permeability of free space.

3.5. Fault Location

For the analyzed system, the length of the transmission line and the velocity of propagation are assumed to be known. This analysis is performed based on two sets of sensors located at opposite ends of the transmission line as shown in Figure 3-13.

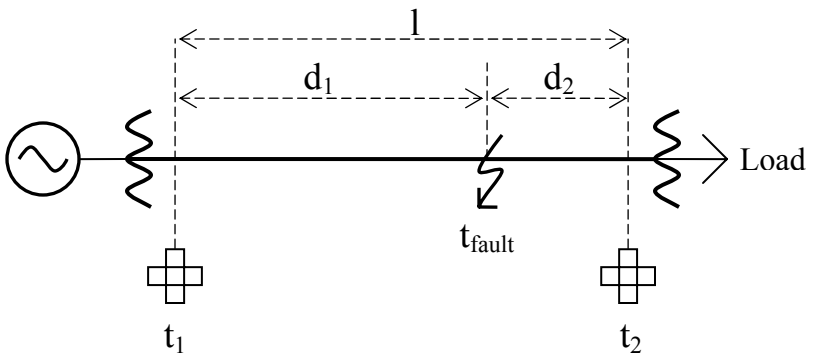


Figure 3-13 – Distances and times used in fault location

Although the velocity of propagation in the transmission line can be described using the Telegrapher Equations[1] as

$$u = \frac{1}{\sqrt{LC}} \tag{56}$$

for a lossless transmission line, where u is the velocity of propagation and L and C are the inductance and capacitance of the transmission line per unit length, respectively, this method of determining the velocity is somewhat cumbersome in this situation since L and C can be difficult to measure for a multi-conductor transmission line. Since the velocity of propagation in an unfaulted transmission line can also be described as

$$u = \frac{l}{t_{trans}} \tag{57}$$

where t_{trans} is the time for an impulse at one end of the transmission line to reach the other end, the velocity of propagation u can be measured without directly knowing the line capacitances and inductances. It is also known that u can be estimated to be close to the

[1] While a complete review of the Telegrapher Equations is essential to an understanding of one-terminal traveling-wave fault location (since this makes use of the reflection coefficient as well as other aspects of these Equations), they will not be discussed in further detail here since the location method explained in this thesis is a two-terminal method.

speed of light in power transmission and distribution lines; however, an accurate value of the propagation velocity is essential in accurate traveling wave-based fault location.

Once u is known, the distance from each sensor and the actual time at which a fault occurs can be found using the equations

$$d_1 - d_2 = u(t_1 - t_2) \tag{58}$$

and

$$d_1 + d_2 = l \tag{59}$$

where t_1 and t_2 are the times at which faults were detected by each sensor in seconds, d_1 and d_2 are the distances from the fault to each sensor in kilometers, l is the length of the transmission line in kilometers, and u is the velocity of propagation in kilometers per second. These result in

$$d_1 = \frac{1}{2}\left[l + u(t_1 - t_2)\right] \tag{60}$$

$$d_2 = \frac{1}{2}\left[l + u(t_2 - t_1)\right] = l - d_1 \tag{61}$$

$$t_{fault} = t_1 - \frac{d_1}{u} = t_2 - \frac{d_2}{u} \tag{62}$$

where t_{fault} is the estimated time at which the fault actually occurred. The actual fault time t_{fault} is very close to the times at which the fault is detected because u is typically very close to the speed of light.

Since the fault detection will be performed by a microprocessor, the analog to digital conversion sampling rate is an important consideration in the accuracy of fault detection. The sampling rate is directly related to the maximum accuracy of the algorithm since it determines the minimum measurable difference in fault detection times. As a

result, the sampling rate must be fairly high to obtain dependable accuracy. The difference in calculated fault location when a calculated fault time changes by a single time step is given by

$$\Delta_{step} = \left(d_1 + \Delta_{step}\right) - d_1 = \frac{1}{2}\left(l + u\left(\left(t_1 + \frac{1}{SR}\right) - t_2\right)\right) - \frac{1}{2}\left(l + u\left(\left(t_1\right) - t_2\right)\right) \tag{63}$$

where Δ_{step} is the step size, or minimum detectable change in meters, and SR is the sampling rate in samples per second. This can be solved to find that

$$\Delta_{step} = \frac{u}{2\left(SR\right)} \tag{64}$$

The factor of 2 is present since a difference of one sample at one end of the transmission line only makes half the impact of one sample at each end in opposite directions. The step size as a function of sampling rate for the ideal value of u = 3×10^8 m/s is shown in Figure 3-14.

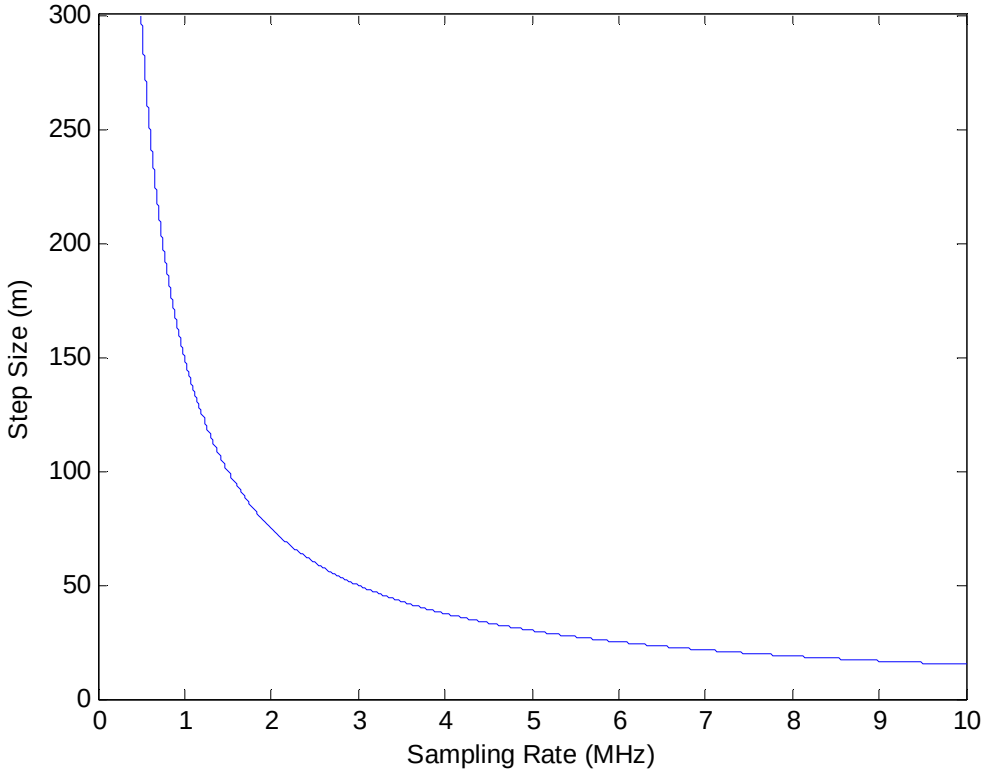


Figure 3-14 – Step size as a function of sampling rate

The value of Δ_{step} is used in determining the maximum accuracy available for any given transmission line and is especially important for shorter transmission lines where the error per length of the transmission line can be significantly affected. Incidentally, the worst-case timing difference between two GPS-synchronized clocks is around 1μs [15], which would, assuming a velocity of propagation of 0.98 times the speed of light, provide a minimum detectable change of approximately 150 meters, assuming insignificant error due to the actual sampling rate. In reality, the sampling rate will also introduce some error, depending on the frequency at which the magnetic fields are sampled. For reasonable accuracy, a sampling rate of greater than 1MHz is desired; this would introduce up to another 150 meters of inaccuracy. However, as technology continues to develop and both sampling rates and the GPS clock speeds are increased, these inaccuracies will be reduced.

4. Analysis Using the Magnetic Field

As previously stated, the most obvious coordinate system to analyze a rotating magnetic field is a polar coordinate system, since any changes in the expected total magnetic field will be detected most easily this way. Four algorithms are presented in this section for the detection of faults while examining the system in polar coordinates. All of them involve detecting if the values of rho or theta have exceeded or gone below a set of expected boundaries or have made a significant and unexpected change.

Since each of these algorithms has a possibility of incorrectly detecting a fault, the results of these algorithms can be analyzed collectively to better determine whether or not a fault has truly occurred. Also, by taking the earliest fault detection times from each algorithm, the microprocessor which is performing this analysis will be able to determine actual fault detection times more correctly in order to perform the fault location more accurately. As a result, this combined analysis using all of these algorithms will provide a reduced number of "false alarms" as well as more accurate fault location.

4.1. Analysis Algorithms

The four algorithms used in this analysis detect the steady-state magnetic field behavior then determine any deviation from it. The first algorithm estimates the ellipse formed by the magnetic field then detects any significant deviations from this locus. The next algorithm compares the present value of rho to the value detected a fraction of a cycle before it and determines if too significant of a change has taken place. The third algorithm detects the maximum change in rho between data points every quarter cycle and determines if the change in rho between the last two data points has exceeded a

multiple of this maximum. Finally, the fourth algorithm detects the maximum and minimum changes in theta every half cycle and determines if the change in theta between the last two data points is significantly higher than the calculated maximum or significantly lower than the calculated minimum.

4.1.1. The "Expected Ellipse" Algorithm

Since the magnetic field will typically form an ellipse in steady state, the simplest way for a microprocessor to determine if there is a fault is to sense if the magnetic field intensities significantly change from the elliptical pattern. There are several ways to perform such an analysis. One is to approximate the shape of the ellipse and determine, once a mostly constant ellipse has been found, if and when the instantaneous magnetic field value deviates from that ellipse. Such a deviation from a constant ellipse is shown below in Figure 4-1.

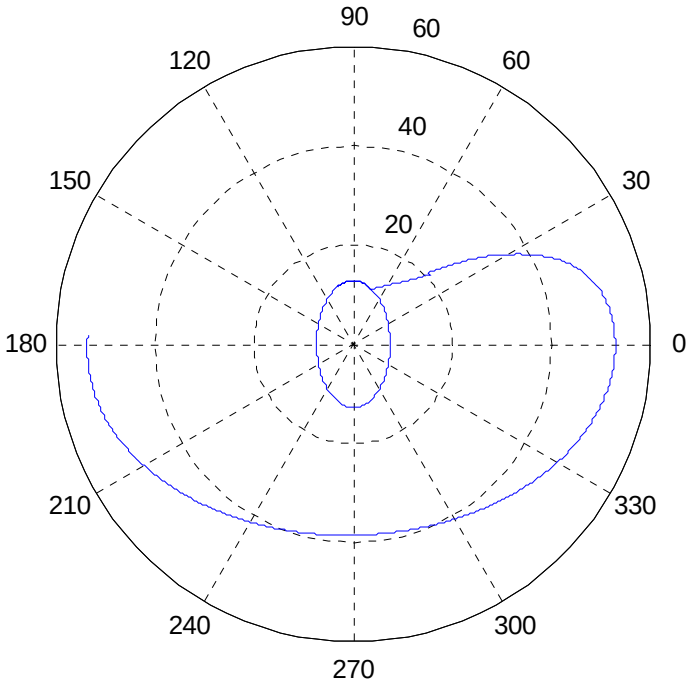


Figure 4-1 – Constant ellipse with a sudden change due to a fault

In order to analyze this rotating field in this way, which will be referred to as the "expected ellipse" algorithm, the following steps are performed:

1. The average maximum and minimum magnetic field intensities and any angle of rotation of the field are determined.
2. An ellipse approximating the rotating magnetic field is generated from this information.
3. Ellipses for the minimum and maximum allowable values of the magnetic field intensity based on allowable percentage deviation from the average must be created from the approximation; these are used to detect any sort of abnormal behavior.

Similar to the Cartesian coordinate ellipse discussed in Section 3.1, the polar coordinate ellipse which can be generated from the information about the maximum rho, minimum rho, and shifted angle is defined by

$$\rho = \frac{\rho_{\max}\rho_{\min}}{\sqrt{\rho_{\max}^2 \sin^2\left(\theta - \theta_{shift}\right) + \rho_{\min}^2 \cos^2\left(\theta - \theta_{shift}\right)}} \tag{65}$$

where ρ is the predicted value of rho for any given value of θ (based on the value of θ for any data point), ρ_{max} and ρ_{min} are the detected maximum and minimum values of rho, and θ_{shift} is the detected rotational shift of the ellipse. Once this ellipse is constructed, it is used in determining allowable maximum and minimum magnetic fields.

An example elliptical rotating magnetic field with several boundaries of allowable values of rho for given values of theta is shown below in Figure 4-2; the actual magnetic field is shown in bold. While values of ρ_{max} and ρ_{min} are indicated, it should be noted that values of ρ_{max} and ρ_{min} also occur at the points 180° around the ellipse from the indicated points. The boundary which is significantly larger than the actual ellipse (the "maximum

boundary") is used for determining if a fault definitely occurred in contrast to the possibility of a fault occurring. This is described further in Sections 4.2.2 and 4.2.3.

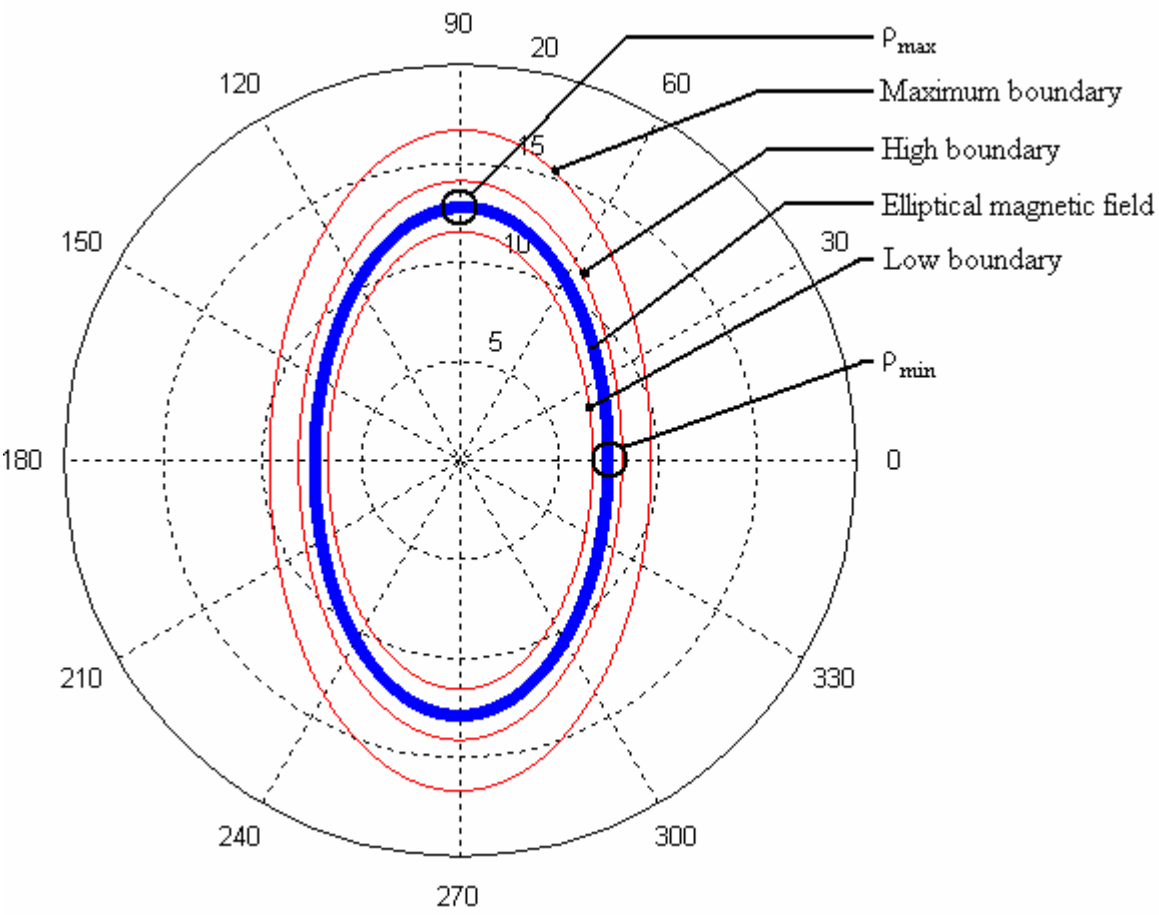


Figure 4-2 – Elliptical rotating magnetic field with boundaries for the "expected ellipse" algorithm

This method of analysis is effective if the maximum and minimum values of rho can be determined accurately. However, if the steady-state currents are distorted, both the maximums and minimums and the angles at which they are detected could be affected. This effect can be reduced by adding an analog or digital filter to the detection device, but there is still a chance of the detection of maximums and minimums being slightly incorrect. Even if these values are close to correct, there is a chance of either a "false alarm" or a fault not being detected with this algorithm alone. In order to decrease the

number of these false positives, a different algorithm can be used in conjunction with the "expected ellipse" algorithm.

4.1.2. The "Previous Value" Algorithm

Since the values of rho in the polar coordinate system do not change significantly over a very short time step for a transmission line that is relatively well balanced, each value of rho can be compared against a value that occurred shortly before it to detect sudden changes. In a sense, this effectively compares the magnetic field against rotated and scaled versions of the same magnetic field. An example of an elliptical rotating magnetic field along with some boundaries generated for this "previous value algorithm" is shown in Figure 4-3.

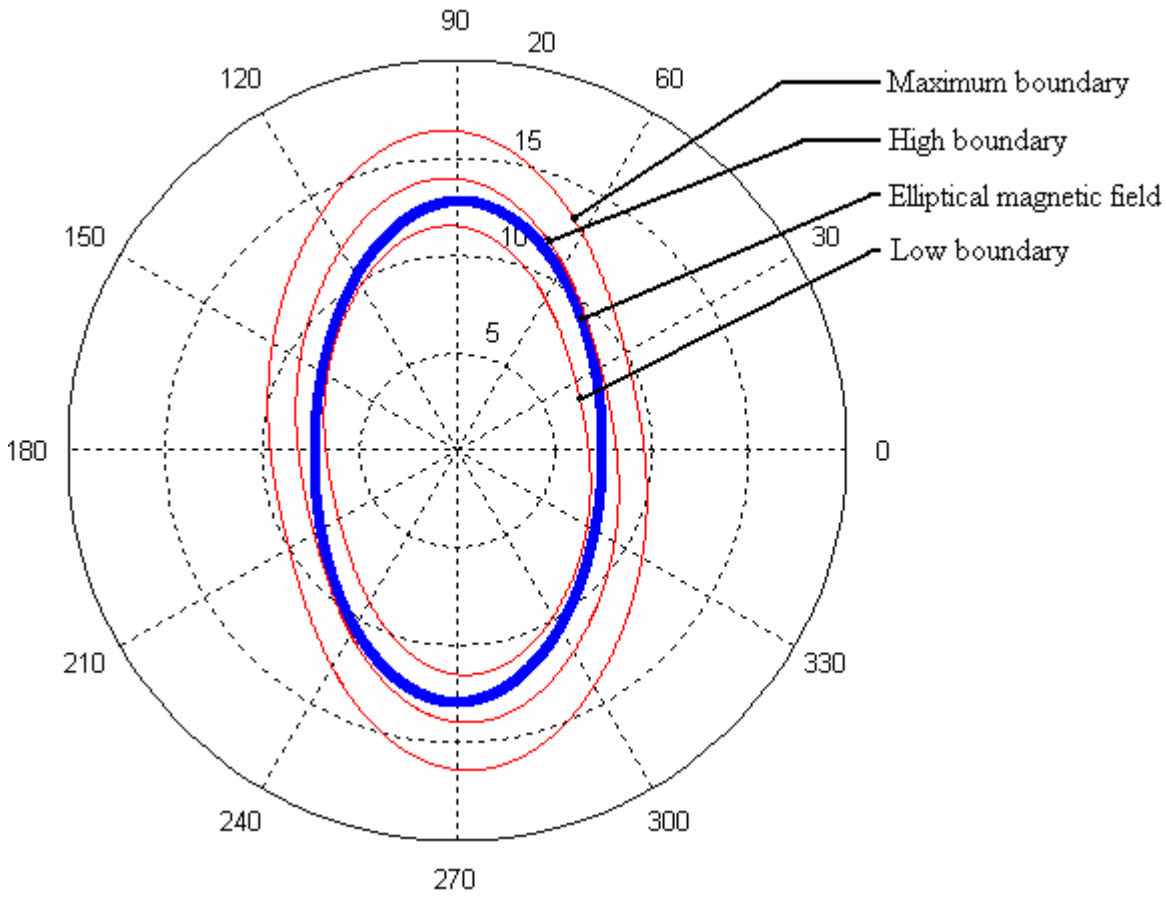


Figure 4-3 – Elliptical rotating magnetic field with boundaries for the "previous value" algorithm

This "previous value" algorithm is especially useful in cases where the magnetic field is not exactly an ellipse and thus cannot be accurately monitored with the "expected ellipse" algorithm. The use of two detection algorithms in conjunction with each other can reduce incorrect fault detections. For example, if the system is fairly imbalanced, faults will be more likely to be incorrectly detected with the "previous value" algorithm, while the "expected ellipse" algorithm will not have as much of a problem with this. Similarly, if harmonics are seen by the sensors and are not properly filtered, the "expected ellipse" algorithm will be much more likely to detect a fault incorrectly while the "previous value" algorithm will not. A magnetic field with several unfiltered arbitrarily phase-shifted harmonics is presented below. Figure 4-4 shows this magnetic

field with boundaries based on the “previous value” algorithm while Figure 4-5 shows the same magnetic field with boundaries based on the “expected ellipse” algorithm. It is clear that the “expected ellipse” algorithm will incorrectly detect faults in this situation. Ideally, all of the harmonics would be filtered prior to analysis, but complete filtering would have negative impacts on the system in other ways including making faults harder to detect. As a result, it cannot be assumed that the magnetic field will be a perfect ellipse. Thus, the fault detection results of the “expected ellipse” algorithm are combined with the results of the “previous value” algorithm in order to better determine if a fault has truly occurred. This example reinforces the idea that performing an analysis of the magnetic field using multiple algorithms in conjunction with one another can reduce the number of incorrect fault detections if the results from each algorithm are compared against those from the other algorithms.

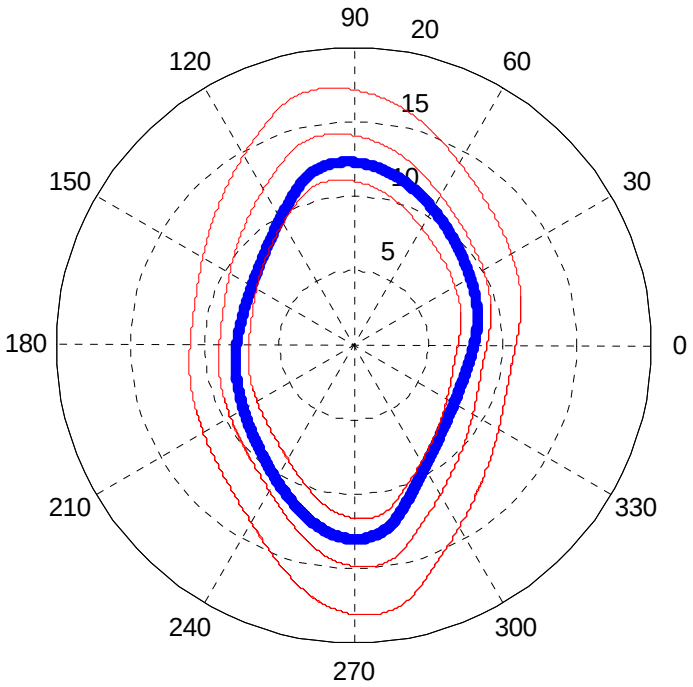


Figure 4-4 – Magnetic field with harmonics, monitored by “previous value” algorithm

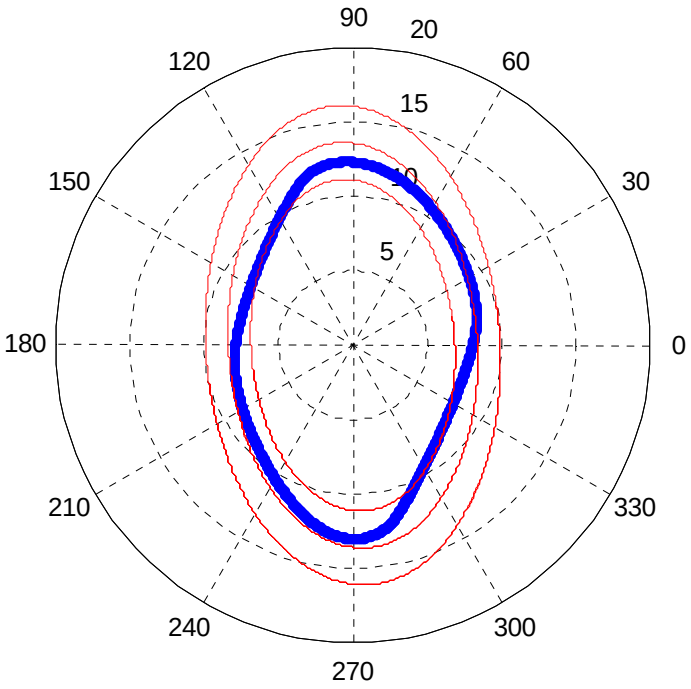


Figure 4-5 – Magnetic field with harmonics, monitored by "expected ellipse" algorithm

4.1.3. The "Delta Rho" Algorithm

As stated before, when a fault is detected with multiple algorithms, the fault location accuracy can be increased by using the earliest fault detection times at each end of the transmission line. Despite this increased accuracy, the time at which the fault is detected with the "expected ellipse" and "previous value" algorithms is not exactly the time at which the fault propagated to the end of the transmission line due to the space provided between the actual magnetic field and the allowable boundaries. There needs to be this small region of allowable variation for each method in order to reduce incorrect fault detections. However, this makes high impedance faults very difficult to detect. In order to increase the accuracy even more and to improve high impedance fault detection, a third algorithm, which will be referred to as the "delta rho" algorithm, is useful.

This algorithm requires less analysis from the microprocessor than the other algorithms, since it simply measures each change in rho between samples against a multiple of the highest change in rho for the unfaulted system. The highest change in rho is assumed to occur halfway between the minimum and maximum values of rho from the "expected ellipse" algorithm for an elliptical magnetic field due to the relationship between the zero crossing of a sine wave and the peak of its derivative which is a cosine. The values of rho and the absolute value of the change in rho for an unfaulted system are shown below in Figure 4-6.

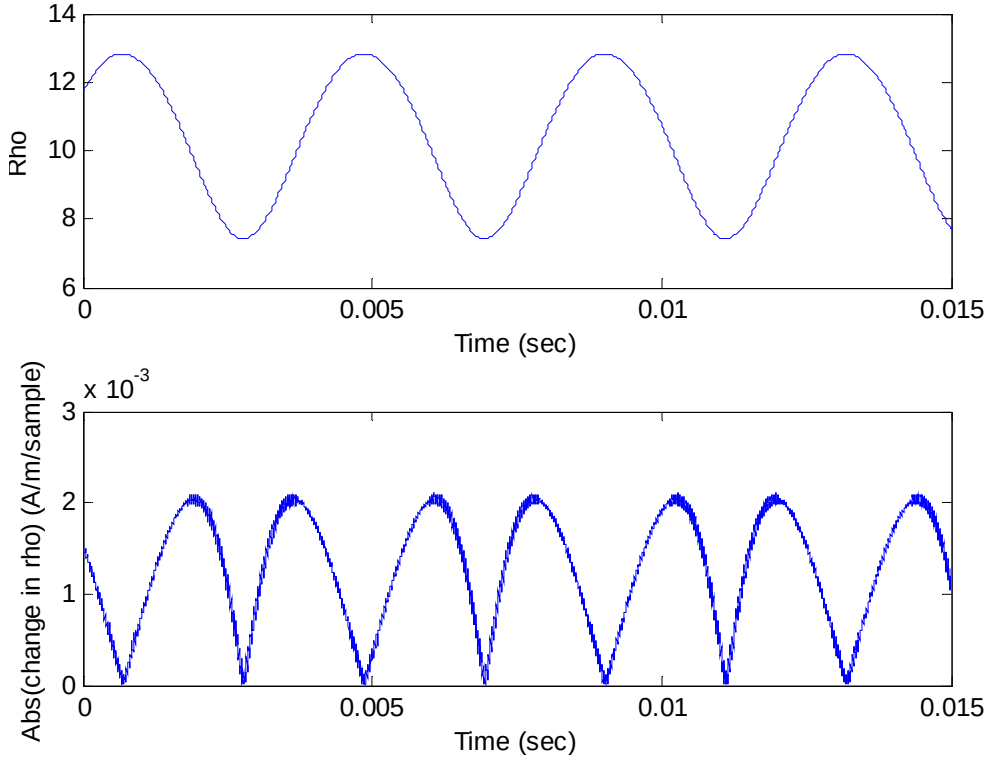


Figure 4-6 – Rho and the absolute value of the change in rho under normal operating conditions

The absolute value of the change in rho is simply calculated as

$$\left|\Delta_{\rho}\right| = \left|\rho_{t} - \rho_{t-1}\right| \tag{66}$$

where the absolute value is used since the maximum allowable magnitude of the change in rho is independent of direction of change. If the change in rho between any two samples is greater than this multiple of the maximum value, there has possibly been a fault. This method will detect a fault time before either of the previously described algorithms and in some sense is most similar to the way a human would determine a fault time based on a visual examination of a plot of the magnetic field.

As an example of this algorithm, the plot of the magnetic field during a three-phase fault is shown in Figure 4-7. It is clear that this is a fault, but the exact time at which the fault caused a change in the magnetic field would be difficult to detect with either the "expected ellipse" algorithm or the "previous value" algorithm. Figure 4-8 shows the values of rho just prior to and during the fault. Again, it is clear that there is a change, but the exact time might be difficult to determine using boundaries. However, it is clear from Figure 4-9 that, once the change in rho is examined rather than the values of rho, the fault can clearly be detected at a specific time where the change in rho increases significantly.

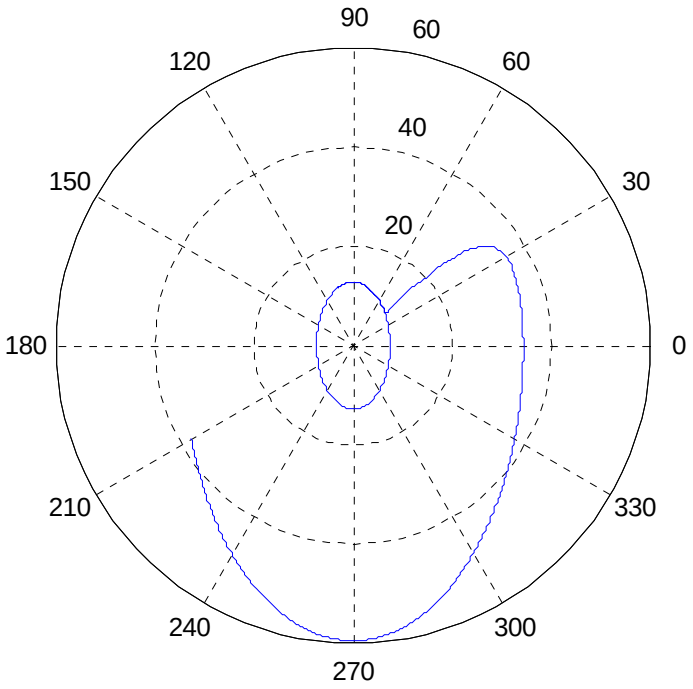


Figure 4-7 – Magnetic field during a three-phase fault

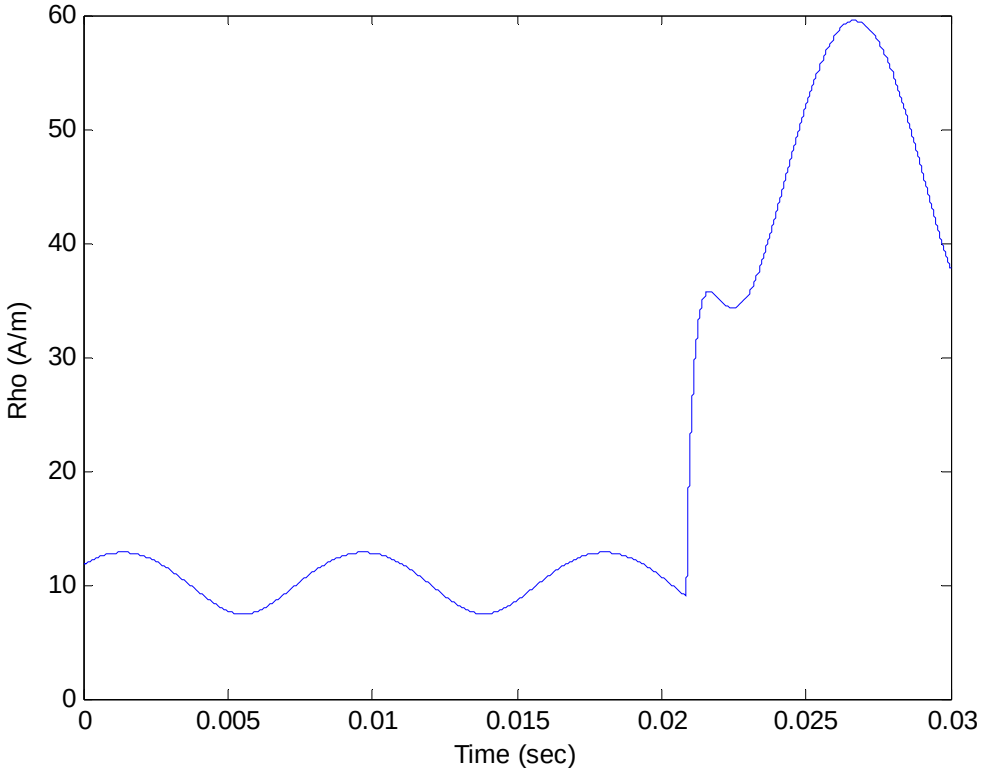


Figure 4-8 – Rho during a three-phase fault

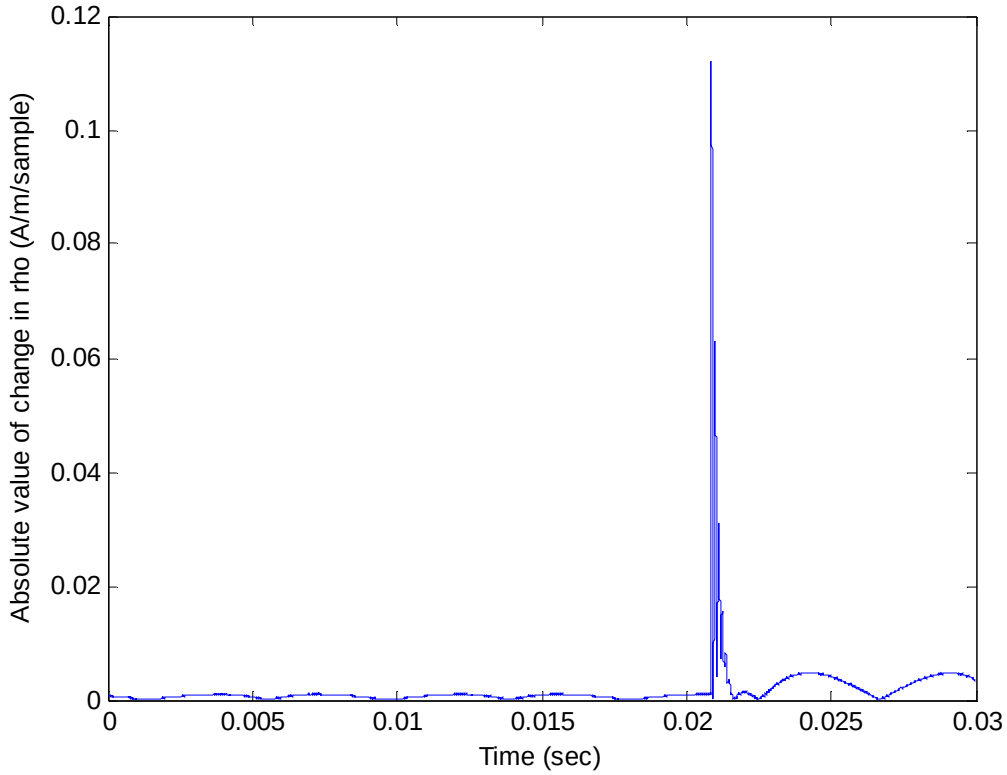


Figure 4-9 – Absolute value of the change in rho during a three-phase fault

The downside of the "delta rho" algorithm is that even a trace of noise can cause an incorrect fault detection. An example of a rotating elliptical magnetic field with some noise added is shown in Figure 4-10; the absolute value of the change in rho for this system is shown in Figure 4-11. Even though the noise does not appear to be very significant when compared to the values of rho as seen in the ellipse, the changes in rho due to the noise are easily enough to cause the delta rho algorithm to incorrectly detect a fault.

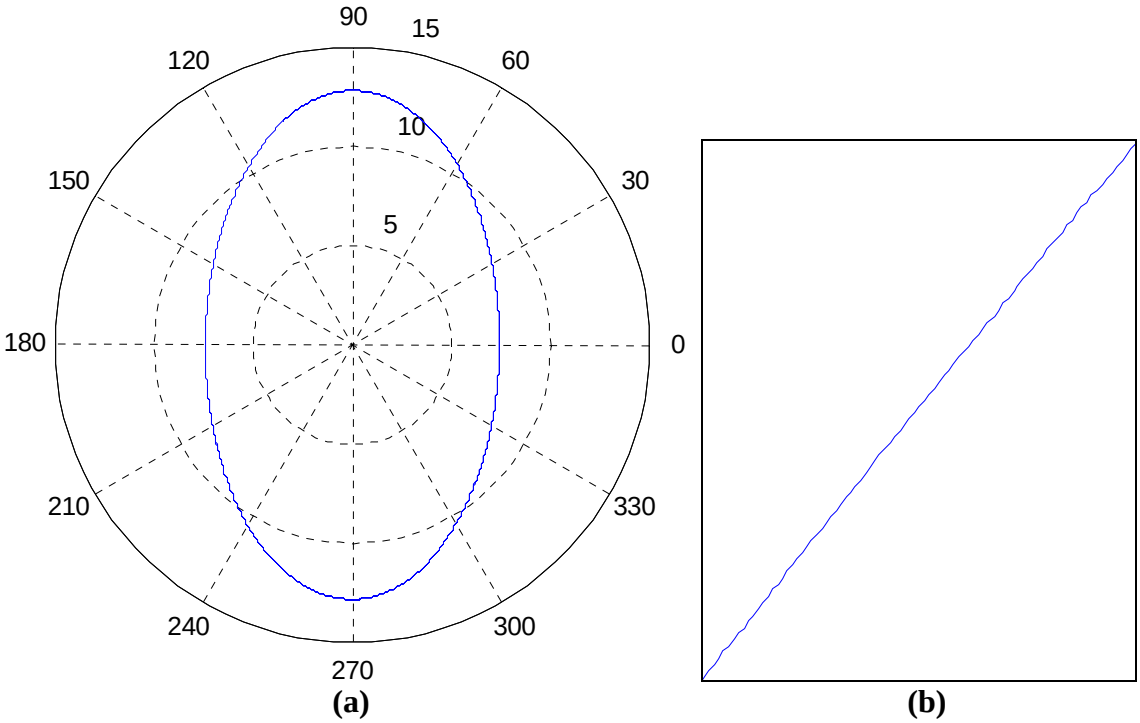


Figure 4-10 – Rotating elliptical magnetic field under normal conditions with noise added
(a) – Full ellipse; (b) – Detail of noise (window is approximately 0.25 A/m in each direction)

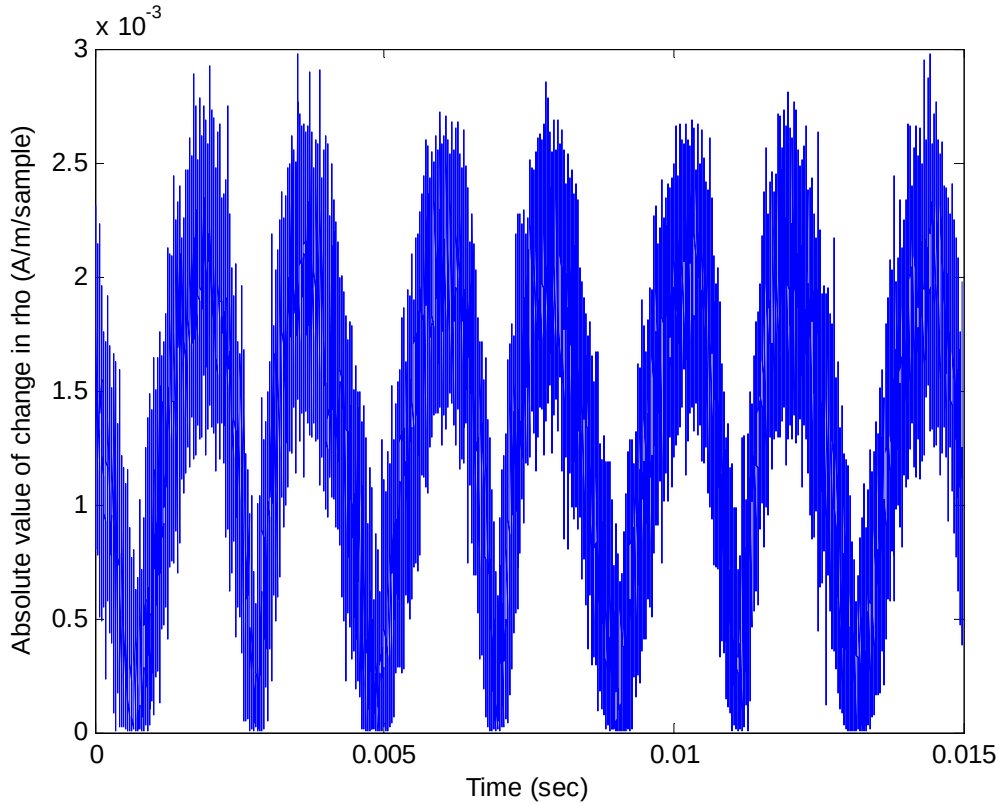


Figure 4-11 – Absolute value of the change in rho under normal conditions with noise added

In order to minimize this risk of incorrect fault detection due to noise, the maximum change in rho is not taken solely from the change in rho over the single time step which would typically create the greatest change, but from an average of the changes per time step over a short period of time. This will significantly reduce the effects of noise, since the noise will be taken into account in this average measurement. There are no negative effects due to this modification under noiseless conditions; under noisy conditions, this change will make the "delta rho" algorithm a bit less likely to detect a fault, but this is clearly superior to the possibility that the algorithm will indicate a fault every time it detects a significant amount of noise.

Since this sudden change in rho can occur due to extreme noise even with this modification in place, the "delta rho" algorithm is more likely to incorrectly indicate faults than the previously discussed algorithms. However, if the time at which a fault is detected with the other algorithms is close to the time at which the "delta rho" algorithm detects a fault, the time from the "delta rho" detection is compared to the other times and is used in determining the fault location, thus providing more accurate location of the fault. Also, since this algorithm detects high impedance faults better than the other algorithms, the fault times related to this algorithm that do not correlate with the fault times of other algorithms are stored separately. In the case that a fault is later found to have occurred, this information can then be used to determine the fault location.

4.1.4. The "Delta Theta" Algorithm

The algorithms described above will detect and locate most types of faults quite well; however, line to line faults, especially those that occur when the currents in the

faulted conductors are at near-equal values, are still problematic with the above three algorithms alone. As a result, another method must be added in order to detect this type of fault. The most distinguishing characteristic of the initial detection of a line to line fault is a rapid change in the value of the polar angle theta (θ). As a result, the most logical detection algorithm to add is one which detects sudden changes in the value of theta. This algorithm will be referred to as the "delta theta" algorithm.

The change in theta per time step is simply defined as

$$\Delta_\theta = \theta_t - \theta_{t-1} \tag{67}$$

The allowable change in theta per time step is between a value slightly higher than the maximum detected change and a value slightly lower than the minimum detected change; if the magnetic field goes beyond either of these boundaries, a fault has most likely occurred. No absolute value is used with this algorithm since a change in the direction in which theta is changing indicates a fault (whereas rho remaining the same or switching from increasing to decreasing is expected under normal operating conditions).

Just like rho, theta changes at different rates throughout the ellipse. The maximum change in theta per time step is at the minimum value of rho, and the minimum change is at the maximum value of rho. This is shown in Figure 4-12; it is important to note that in this Figure the "maximum" value of theta is actually a minimum since theta is always negative (constantly decreasing) for the particular situation. Since the "delta theta" algorithm is prone to the effects of noise in the same way as the "delta rho" algorithm, the same averaging method is used in calculating the maximum and minimum allowable values in order to reduce the number of incorrect fault detections.

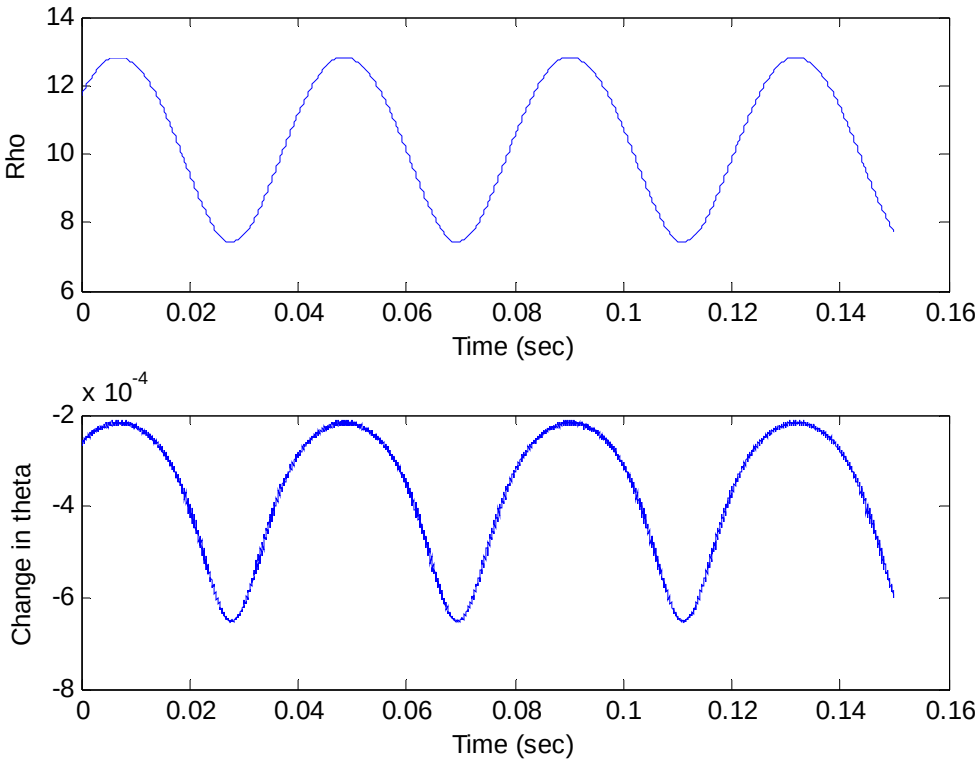


Figure 4-12 – Rho and delta theta compared under normal operating conditions

The magnetic field of a line to line fault is shown in Figure 4-13; Figure 4-14 shows the value of theta up to and during the fault. There is clearly an abrupt change in the value of theta when the magnetic field changes its pattern due to the fault current, but the exact time of this change is difficult to determine directly from theta. In contrast, the change in theta is shown in Figure 4-15. The sudden change due to the fault can be easily be located using this information.

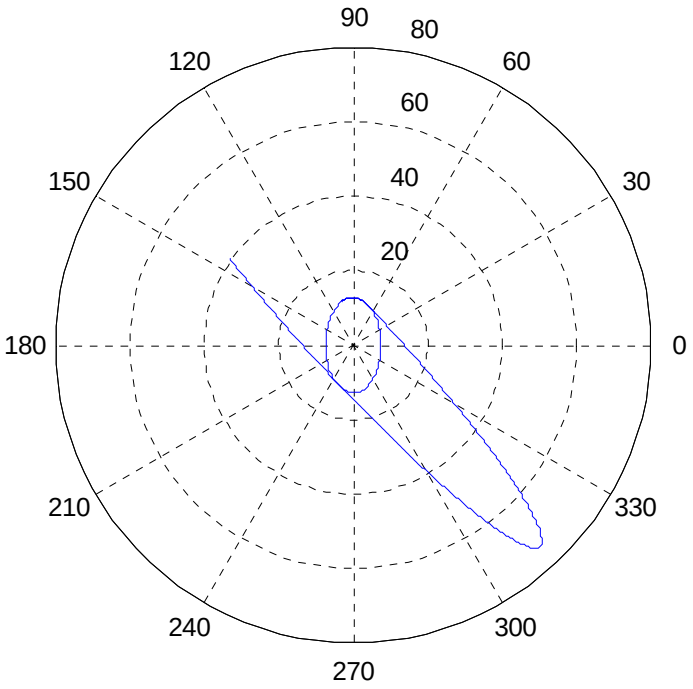


Figure 4-13 – Magnetic field during a line to line fault

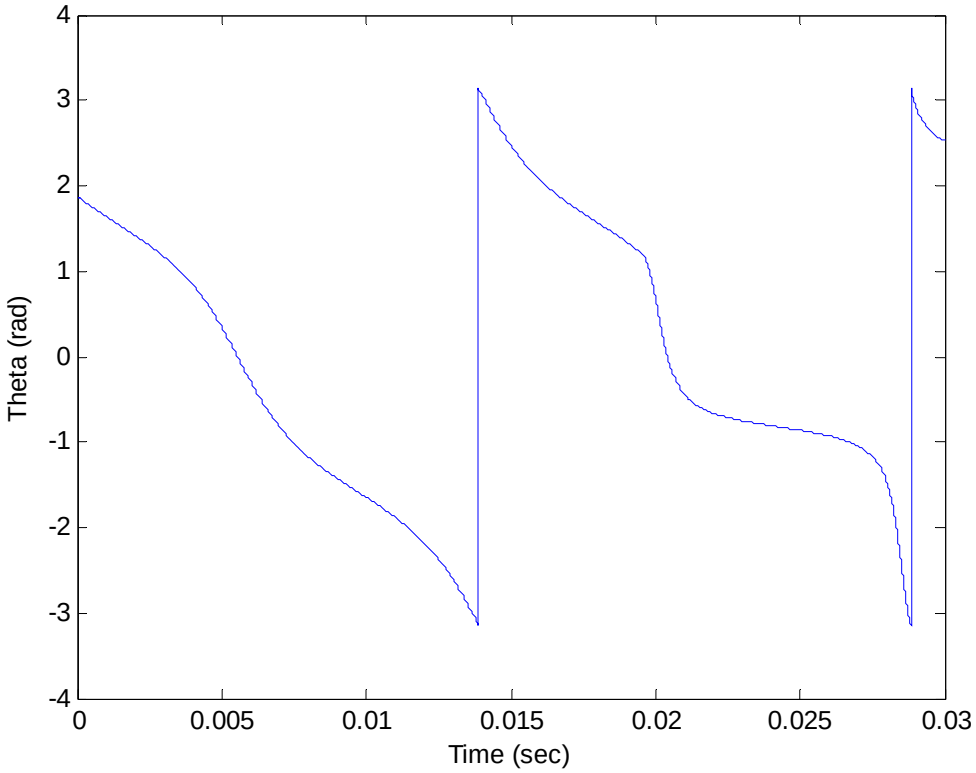


Figure 4-14 – Theta during a line to line fault

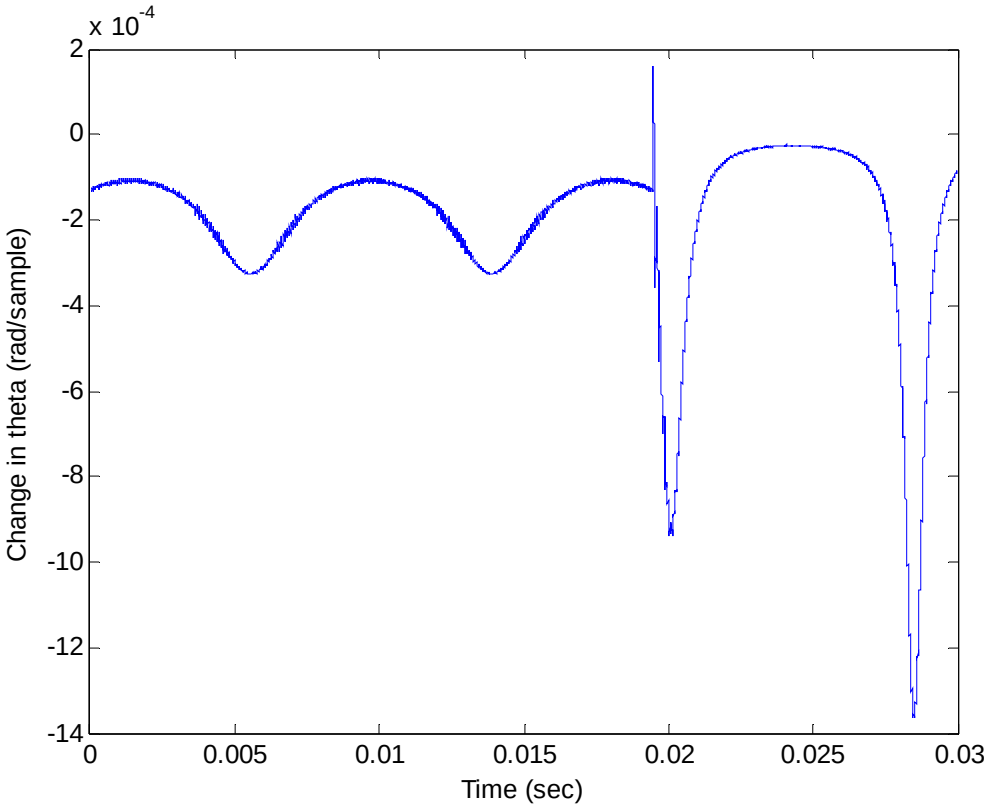


Figure 4-15 – Change in theta during a line to line fault

4.1.5. Fault Detection

Each algorithm detects faults independently; once faults have been detected by one or more algorithms, the fault location is performed. If there is some kind of abnormal behavior (a field intensity that goes outside of a minimum or maximum, or a change greater than allowed by the "delta rho" algorithm), the detection system will perform the following operations:

1. Record the time at which this occurred.
2. If some abnormality is recorded at both ends of the transmission line, then the location of the fault is computed based on the difference in detection times.

3. If the abnormality is recorded at only one end of the transmission line, then the possibility that an error might have occurred is recorded in the microprocessor memory.

4.2. Implementation of the Algorithm

While the collective algorithm is not particularly difficult in concept, the implementation becomes somewhat complicated since the fields in question are not necessarily limited to ellipses in their shapes as has been noted in previous sections. This necessitates the addition of several steps to the algorithm to prevent "false alarms" and to increase the accuracy of the algorithm as much as possible. It is essential to note at this point that all calculations are done identically and independently for both ends of the transmission line, and the only time these calculations interact is when a potential fault has been detected.

This particular implementation of the algorithm is done in a time loop, similarly to how an actual device would function, with the only time-independent element being a readout of possible fault information at the end of the code. The analyses for each algorithm can be run in parallel if necessary to increase the processing speed of the complete analysis. Figure 4-16 shows the implementation of the collective algorithm. The complete MATLAB code can be found in Appendix C.

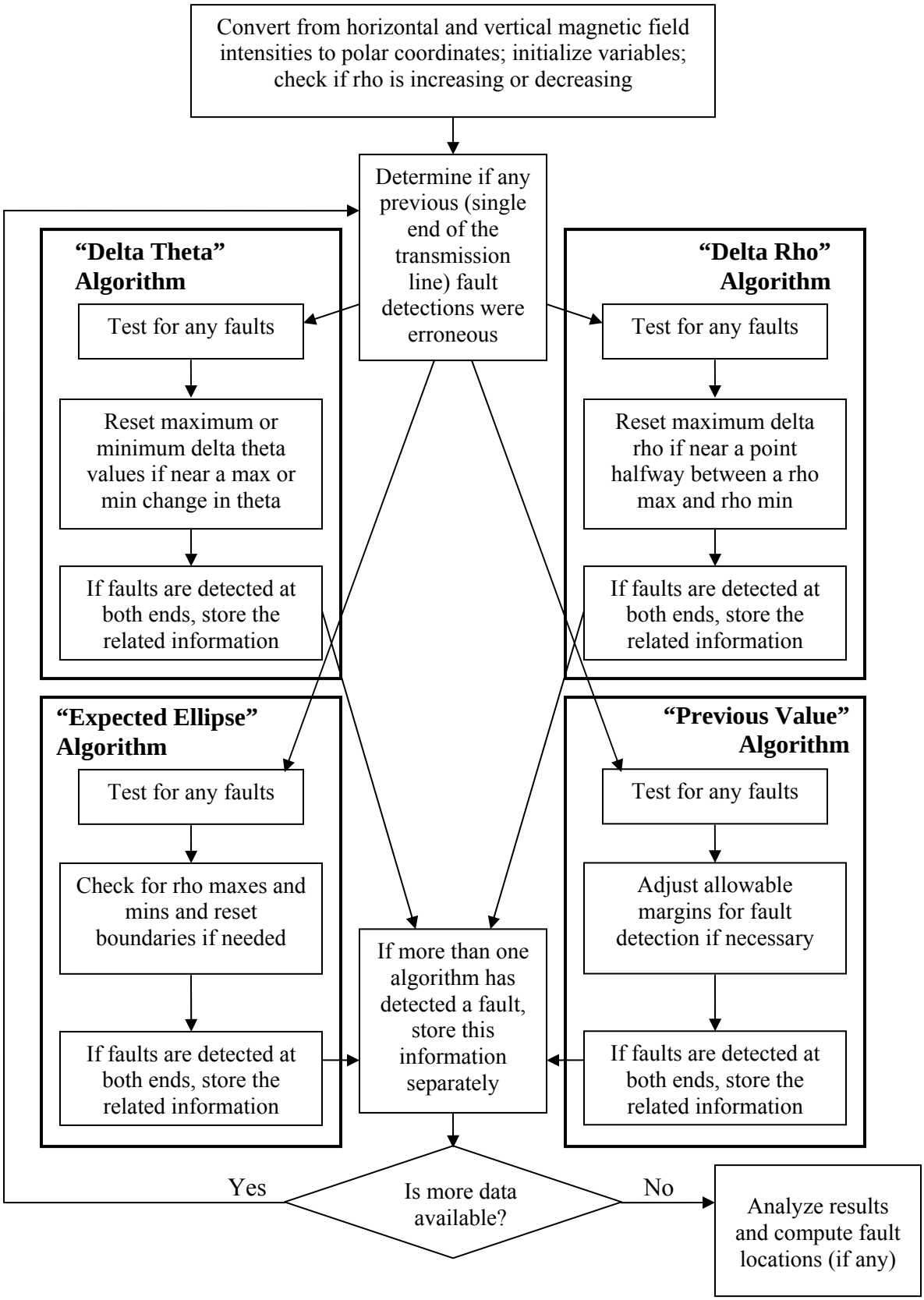


Figure 4-16 – Fault Detection Algorithm

4.2.1. Initialization

The process begins by receiving the inputs – specifically, currents in the conductors, the transmission line geometry and length, sampling rates, the velocity of propagation, the time from which the analysis starts in case it is referenced to a time external to the code, and the time to wait for the transmission line to be re-energized before resuming analysis. The currents are converted into the horizontal and vertical magnetic fields using Equations (47) and (48) and the information which was provided by the user. These values are then converted into the values of rho and theta for all values of time. A MATLAB function called atan2 is used in this implementation to take the inverse tangent to find theta; this takes negative H_x into account and determines in which quadrant theta should be located. Rho is calculated using Equation (51).

These calculations are followed by an initialization of a large number of other variables which are used at various points in the analysis. After the basic variable initialization, all of the variables related to the minimum and maximum rho values and the angle of rotation from a standard ellipse are found by calling two functions. These functions have been named “checkdir” and “minmaxrho” in the code; they find whether rho is increasing or decreasing and the minimum and maximum values of rho, respectively. These values are used in the analysis for the “expected ellipse” algorithm.

The check for whether rho is increasing or decreasing is essential in determining the minimum and maximum values of rho; it is implemented separately from the function that determines the minimum and maximum values to simplify function calls in the code and to make debugging easier. This function in and of itself is rather simple. It checks the

rho value for the present time value then compares it to the value for the previous time. This determines if rho is increasing or decreasing.

Finding the minimum and maximum values of rho is somewhat more complicated. At the beginning of the function, all the necessary variables are temporarily set to zero. The function will continue running until all of the necessary values (rho minimums and maximums as well as angles of rotation) have been found, or until the end of the available information has been reached. It is clear that the maximum and minimum values for rho should occur when rho changes from increasing to decreasing or decreasing to increasing, respectively; however, harmonics, noise, or errors in analog to digital conversion may indicate local maximums or minimums that are not the maximum or minimum for the system in this way. In order to remove the effects of these incorrect detections of maximums and minimums, once a maximum or minimum is detected, the system searches through the previous quarter cycle for the highest (or lowest, in the case of a minimum) values of rho and uses this value. In order to prevent faults from incorrectly affecting these results, the amount of time between detections of a maximum and a minimum and vice versa is limited. As a result, faults which cause a change in a maximum or minimum will only change the predicted ellipse based on the first changed value due to the fault; any later changes in rho will indicate a fault. The values for the angles of rotation (called “thetashift” in the code) are taken to be the theta values where the maximum rhos are detected. The resulting maximums, minimums, and angles of rotation are then returned to the main program.

4.2.2. Error Checking

At this point, problems from previous time steps are analyzed. Any temporary variables that have expired are cleared. If a fault is detected at one end of the transmission line with any algorithm, the system waits for the maximum possible detection time, defined by a value slightly larger than the length of the transmission line divided by the velocity of propagation. This is the maximum time that could occur between detection at one end of the transmission line and detection at the other for an actual fault. If the time is exceeded, this means that the indication at one end of the transmission line was an incorrect detection or that a fault occurred but that one end of the transmission line did not see any evidence of this fault. The latter situation is likely to occur for high impedance faults or for faults near the zero-crossing of the current waveform in the faulted phase. If the fault that was detected was due to an increase above the maximum allowable rho value, a "problem time" is recorded, since the maximum rho should theoretically never be exceeded except in the case of a fault; thus, for the maximum to be exceeded at one end of the transmission line without any fault detection at the other, it means that there has been some error. Otherwise, the fault detection is assumed to be an incorrect detection. Either way, all flags indicating a fault for the algorithm which detected this fault are cleared after this maximum time in order to reset the system as quickly as possible.

Additionally, if the "delta rho" algorithm has been indicating faults repeatedly without the other algorithms detecting any faults, this could mean that there is excessive noise in the system for some reason. As faults are detected by the "delta rho" algorithm, a variable is set to count until a quarter of a cycle has passed. If at least three faults have

been detected by the “delta rho” algorithm within this period of time, the maximum percentage allowable above the highest change in rho in the system is doubled. This will prevent repetitive tripping due to the “delta rho” algorithm. Since it is assumed that the significant amount of noise required to cause this algorithm to detect a fault is a temporary condition, the original maximum allowable percentage above the highest change in rho is restored after a fault occurs or after an extended period of time.

4.2.3. Fault Detection

After finding all the information necessary for analysis, the fault detection can begin. Fault detection is performed before resetting any of the algorithm variables since a change in a variable such as the maximum detected value of rho may be due to a fault. Thus, the fault detection is done first to reduce any impact that these variable changes may have on the detection process.

The first attempt at fault detection is using the “delta rho” algorithm. If the change in rho between the current time step and the previous time step is greater than a multiple of the largest change in rho that was recently detected, this algorithm indicates a fault.

After this, the present values are tested with the “delta theta” algorithm. If the value of theta has changed more than a multiple of the highest detected change or a fraction of the lowest detected change, a fault is indicated. It is important to note that there is a break in the values of theta since the inverse tangent function (which is used in converting the horizontal and vertical magnetic field values into polar coordinates of rho and theta) only returns values in the range $[-\pi,\pi]$ or $[0,2\pi]$ (after correction for points in quadrants II and III). As a result, a modification must be made as the value of theta goes beyond these boundaries since this will indicate a change in theta of nearly $\pm 2\pi$. If the

value of theta has just made this change in either direction during the present time step, the change in theta is calculated by subtracting 2π from the higher of the two values. As a result, the actual change in theta is calculated.

The code continues with several checks for faults using the "expected ellipse" and "previous value" algorithms. While a simple check for an increase in rho beyond the maximum allowable value at one end of a line and a decrease beyond the minimum allowable value at the other end might find faults, the fault locations which are determined by the algorithm will not necessarily be very accurate.

The analysis begins by testing for an increase beyond the maximum rho at either end of the transmission line. If either end satisfies this condition and has not detected a fault recently, the time of the fault and other information will be recorded along with a flag to indicate the cause of the fault detection at that end of the transmission line.

After comparing the current value of rho against the maximum rho, the current rho is compared to a different value of rho above the expected value (referred to as the "high" value of rho) which is analyzed similarly to the maximum value. Additionally, since the high value of rho must be exceeded in order to reach the maximum value of rho, if one end of the transmission line has been flagged for exceeding the high value of rho, this flag can be cleared and replaced with another flag if the maximum is exceeded.

The value of rho is also compared to the minimum allowable value of rho. If the present value of rho is less than the minimum allowable rho at either end of the transmission line, that end of the transmission line is flagged. Additionally, if the value of rho drops below the minimum then suddenly rises above the minimum at the other side of

the ellipse, a different kind of flag is stored since this type of behavior is more likely to indicate a fault than simply a drop below the minimum.

The amount of time that the value of rho is allowed to remain under the minimum value before the algorithm chooses not to store this different flag is identified in the code as “maxbelowtime.” A set of variables called “dropbelow” are defined for each end of the transmission line and for both the “previous value” and “expected ellipse” algorithms. As every time step is analyzed, the “dropbelow” is incremented (starting from zero) for any algorithm which has detected that the value of rho has gone below the minimum allowable value. If this value reaches “maxbelowtime” without the value of rho returning above the minimum allowable value, it is assumed that rho will not be going above the minimum value again and thus this flag for a stronger indication of a fault will not be set.

4.2.4. Resetting Variables

This algorithm continues with code very similar to the “minmaxrho” initialization code described above, except it occurs as the time is passing; as a result, the minimum and maximum values for rho and the angle of rotation are only updated at time values where the condition for a minimum or maximum has occurred.

For the “delta rho” algorithm, the value of the maximum change in rho detected in the system should occur near the angle half way between the maximum value of rho and the minimum value of rho. Simply taking the difference between the value of rho during this time step and the value during the previous time step should produce the maximum change in rho, but if there is noise in the system, this could produce a change in rho much lower than the highest change present in the non-faulted system.

One way to solve this problem is to check the change in rho for every time step and determine this change is greater than the previously defined maximum change in rho. However, this would require a great deal of additional analysis and more importantly, it would be affected by faults, since a fault would simply change the maximum allowable value of rho and thus make it harder to detect further increases in rho due to the same fault.

Instead, this algorithm averages the changes in rho over a short period of time near the angle where the maximum change in rho should occur. This reduces the effect of noise since the average change in rho due to the noise is taken into account. Faults would not significantly affect this value since a fault that occurred a few time steps back would be detected at this point. Additionally, the changes in rho over several time steps are averaged so if a fault is just about to be detected and has made a small change to the current value of rho but not enough to be detected, this small change in rho that has not yet been detected is only one small part of the average.

Similarly to the "delta rho" algorithm, the values for the "delta theta" are initialized as averages. The maximum change in theta is taken to be the average change in theta near the minimum value of rho while the minimum change is taken as the average near the maximum value of rho.

Since the "previous value" algorithm tests the present value of rho against multiples of the rho values from previous time steps, the algorithm is prone to indicating faults incorrectly when rho changes rapidly – for example, in the case of extreme imbalance between the currents in the conductors. The allowable boundary values of the "previous value" algorithm can be modified to prevent this kind of situation while

keeping the boundaries close enough to the allowable values to detect faults as quickly as possible.

Throughout each cycle, the value of rho is compared against four boundaries which are a set percentage smaller than the fault detection boundaries for the "previous value" algorithm. Two of these are slightly above and below the predicted value of rho. Ideally, these boundaries will be exceeded at least for a bit of a cycle; if either of them is not, the allowable value of rho in that direction (either above or below, dependent upon which boundary is not exceeded) is reduced in order to increase the fault detection accuracy of the algorithm. The other two values are much closer to the fault detection boundaries. If either of these is exceeded over the course of a cycle, it means that the algorithm is getting close to detecting faults incorrectly (or that a fault has occurred, but presumably this would also exceed the allowable value and indicate a fault in addition to exceeding these testing values).and a result, the allowable value of rho in the appropriate direction is increased.

4.2.5. Fault Information Storage

If a fault has been detected at one end of the transmission line, the time at which it was detected is stored. Once flags indicating faults have been set at both ends of the transmission line, the system stores the information about the fault times and indication types prior to temporarily suspending its operation. If only one algorithm is indicating a fault and if that algorithm has not exceeded the waiting time mentioned in Section 4.2.2, the system will wait for another algorithm to indicate a fault as well; as previously stated, taking the earliest detection times at each end of the transmission line from two algorithms can provide more accurate fault location. Even if the waiting time has been

exceeded, if only the “delta rho” algorithm or only the “delta theta” algorithm has detected a fault, the system will not suspend its operation as it would for detection with either of the other algorithms or for detection with multiple algorithms. This is done since these two algorithms (the “delta rho” algorithm in particular) are sensitive to noise, and the fault detection should not be suspended based solely on algorithms which could be prone to having this sort of problem. However, if one of these algorithms detects a fault but the waiting time is exceeded, this fault timing information is stored separately from the other fault detection timing information in case it is later determined that a fault actually occurred.

This information is processed at the end of the available time information for the present implementation of the algorithm; in practice, the analysis would be performed as faults are detected since there is no end of data in a real-life situation. The algorithm then pauses for a short period of time (which is provided by the user) to allow the breaker to clear the transmission line. This is to prevent faults from being erroneously detected immediately after resuming analysis. After this waiting time has passed, the system again initializes the variables for its operation (minimum and maximum rho values, etc.), removes the fault-indicating flags, and re-enters the time loop to wait for another fault.

It is essential to consider that, depending on the time delay between the fault detection and resumed operation of the detector, the transmission line may not yet be re-energized when the detector resumes operation. Since the minimum and maximum rho will be determined based on this system, the transmission line being re-energized may also indicate a fault. In practice, however, this fault locator device would likely be

networked with other protection equipment and thus would be aware of the status of the transmission line.

4.2.6. Fault Analysis

After the end of the available data, any faults are analyzed using Equations (60), (61), and (62); prior to this, the information was only stored and was not processed. For each fault, timing information may be available from any of the algorithms which detected it. In order to obtain the most accurate measurement possible, the earliest available detection times for each fault are used in the calculation of the fault location. The location of the faults and the approximated times at which they occurred are then presented along with the information about the type of change that caused the fault to be detected.

If any fault location calculations result in a negative distance, the times resulting in these errors are presented. As described in Section 4.2.2, a bit of leeway is given in waiting between the detection of a fault at one end of the transmission line and the latest expected fault detection time at the other end (at which time the information about the detection of the fault at the first end of the transmission line is discarded). These negative distance calculations may be the result of a fault very close to one end of the transmission line and late fault detection at the other end of the transmission line which falls within this leeway. The resulting negative distance can thus indicate a fault very close to one end of the transmission line.

Additionally, any of the "problem times" – where a fault was clearly detected at one end of the transmission line but nothing was detected at the other end – are shown

after the end of the available data. This information can be used for “debugging” and system analysis.

5. Testing the Algorithm

The full algorithm was tested to determine the accuracy with which it can detect faults. The system was tested for a 115kV transmission system. The geometry of the transmission line and details about its simulation model are presented in Appendix A. Single line to ground faults were used as the main fault for testing since they are by far the most common fault types; the conductor configuration was chosen as a coplanar arrangement. Since the accuracy of calculation is dependent upon the angle of fault incidence, the system was tested for faults at both the zero-crossing of the faulted phase's current, which is the most difficult fault timing to detect, and at the faulted phase's peak current, which is easiest to detect. The testing was based upon a purely resistive fault. The location of the fault was varied linearly along a 20km transmission line for fault resistances of 0.1, 1, and 10 per unit, which are equivalent to approximately 13.225Ω, 132.25Ω, and 1322.5Ω for the 115kV transmission line in question. These values come from

$$Z_{base} = \frac{\left(kV_{LL,base}\right)^2}{MVA_{3\phi,base}} \tag{68}$$

for the given base voltage of 115kV line to line and a chosen base 3-phase power of 100MVA which results in a per unit impedance of 132.25Ω.

The sampling rate used in this testing was 2MHz. For all tests, any frequencies below 60Hz and any harmonics are assumed to have been filtered in a way that does not noticeably affect the fault detection. Additionally, the magnetic sensors are assumed to be able to respond to a change in the magnetic field instantaneously. Also, noise is assumed to be negligible.

The fault simulations were conducted using ATPDraw, a free electric power system analysis program; the resulting data was exported into Microsoft Excel using TOP, an output processor for power system analysis programs. The Excel-format data was then processed by MATLAB using the code in Appendix C. In an actual power system setting, only the actual analysis step (here represented by the MATLAB analysis) would be performed, thus simplifying the process significantly.

The results of these analyses are shown below in Figure 5-1 and Figure 5-2. The fault location error is specified in [3]. It is equal to

$$Error\% = \frac{d_{read} - d_{actual}}{l} \times 100 \tag{69}$$

where d_{read} is the fault location detected by the algorithm, d_{actual} is the actual location of the fault, and l is the total length of the transmission line. If a fault was detected with only one of the algorithms, this detection was not discounted.

It is clear from a comparison between the accuracies of the fault detections for zero-crossing current faults and the fault detections for peak current faults that the fault location is much more reliable for faults at the peak of the current. In fact, the curve in Figure 5-1 (which is the best-case situation for fault location) is independent of fault resistance until the fault resistance becomes larger than 10 per unit (1322.5Ω) which is the resistance for which this Figure is plotted. The faults at the current's zero-crossing were not detected at one end of the transmission line (thus preventing fault location) for fault resistances greater than 1 per unit, which is 132.25Ω. (Note that the algorithm did detect and locate the faults at 10% and 20% of the transmission line length at this impedance, but the accuracy was incredibly poor and thus the points do not appear in Figure 5-2.) For the sake of comparison, ground fault resistances may be as high as 800Ω

"[i]n cases of high ground resistivity and no overhead ground wire" [16]. While this is a relatively high impedance for a ground fault, it is still possible that a fault of this impedance could occur near the zero-crossing of the faulted phase's current and that as a result this algorithm would not be able to detect or locate the fault. This is one of the admitted weaknesses of this algorithm. As a result, this implementation of the fault detector can only be recommended for use with transmission lines that have at least one overhead ground wire.

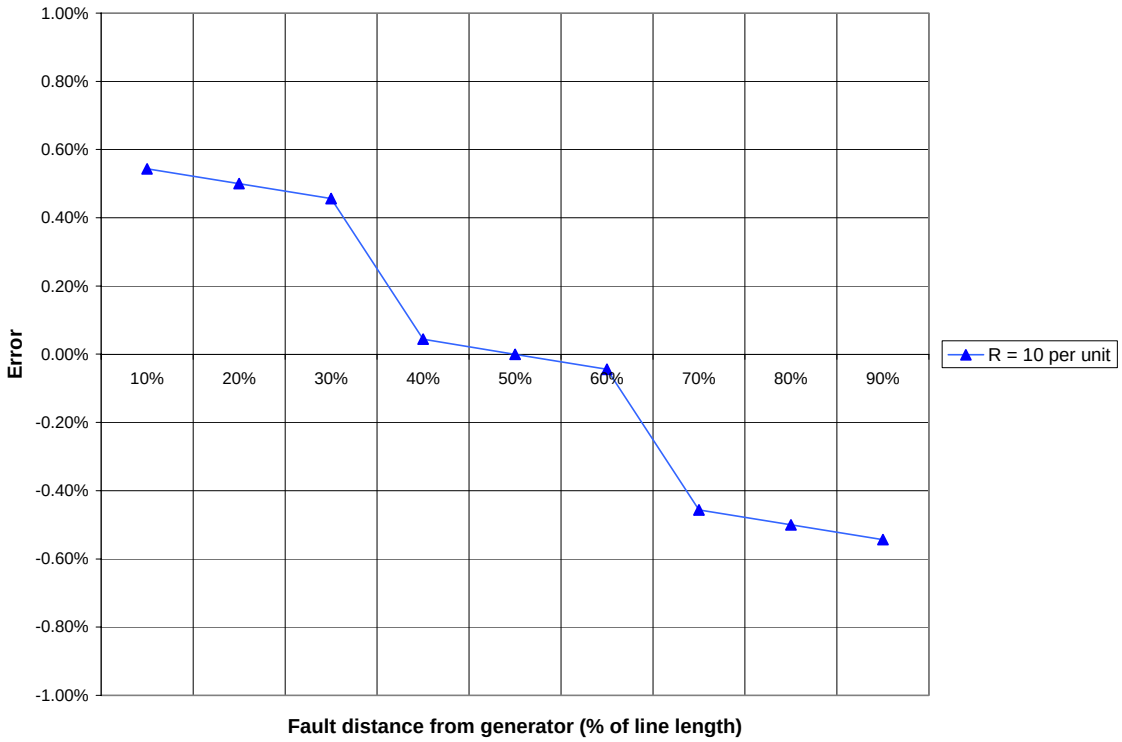


Figure 5-1 – Fault location error for a single line to ground fault when the faulted phase's current is at a maximum

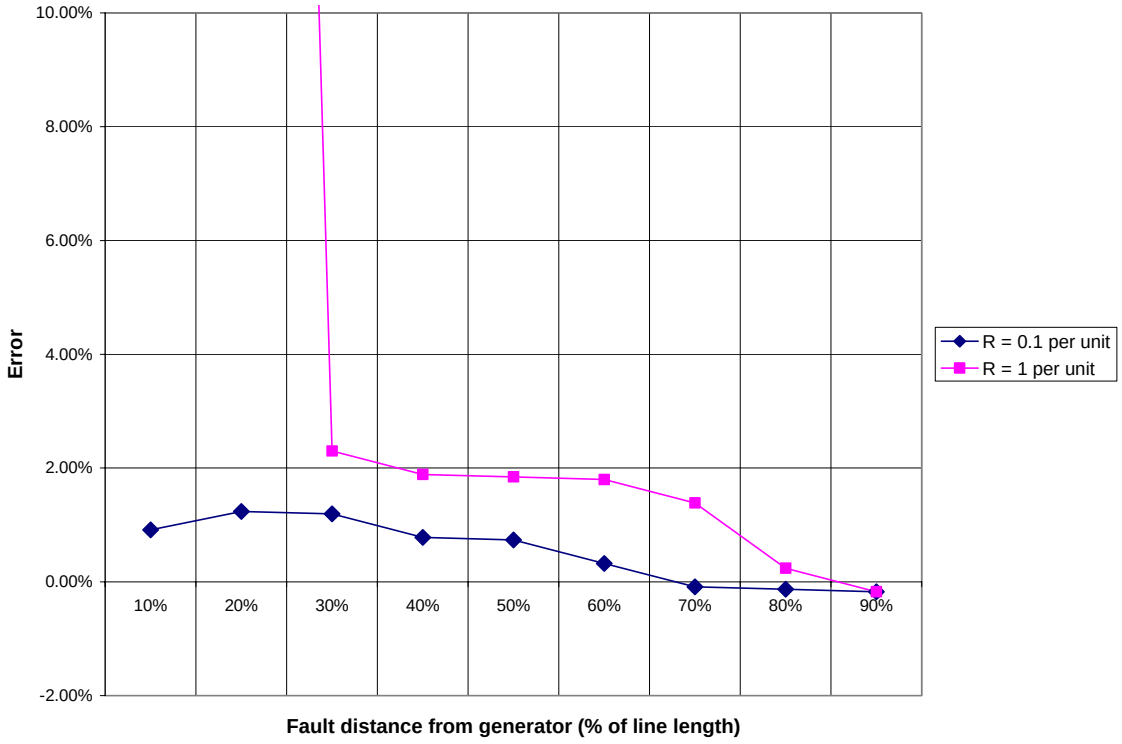


Figure 5-2 – Fault location error for a single line to ground fault when the faulted phase's current is zero

The algorithm was also tested for a range of fault resistances for each major fault type, including single line to ground faults (at both the faulted phase's current maximum and current zero-crossing), line to line faults and line to line to ground faults (when both phases' currents are at identical values), and three phase faults (at one of the phases' current zero-crossings). The timings above are the most difficult portions of the cycle to detect each of these faults, with the exception of the single line to ground fault at the faulted phase's current maximum which was included since it was already tested based on fault location in Figure 5-1. The faults tested were located in the center of the transmission line (10km from each end) in order to minimize the faults appearing to be undetected when the problem is simply a large error. For example, if a fault is very close to one of the ends of the transmission line and the other end does not detect the fault exactly when the traveling wave arrives, the fault will appear to be off of the transmission

line and, based on the way the analysis algorithm is designed, will be seen as an erroneous fault detection.

The results, which are shown below in Figure 5-3, Figure 5-4, and Figure 5-5, indicate extremely accurate fault detection. Aside from the slightly low detectable fault impedance for the worst case of single line to ground faults as previously mentioned, the only fault type where moderate fault impedances seem to cause a problem is the line to line fault. As can be seen in Appendix B, this fault location must rely specifically on detecting sudden changes in theta to produce an accurate result. By decreasing the allowable range of values of delta theta, the maximum detectable fault impedance could be increased; however, this would make the algorithm more likely to incorrectly detect faults due to noise.

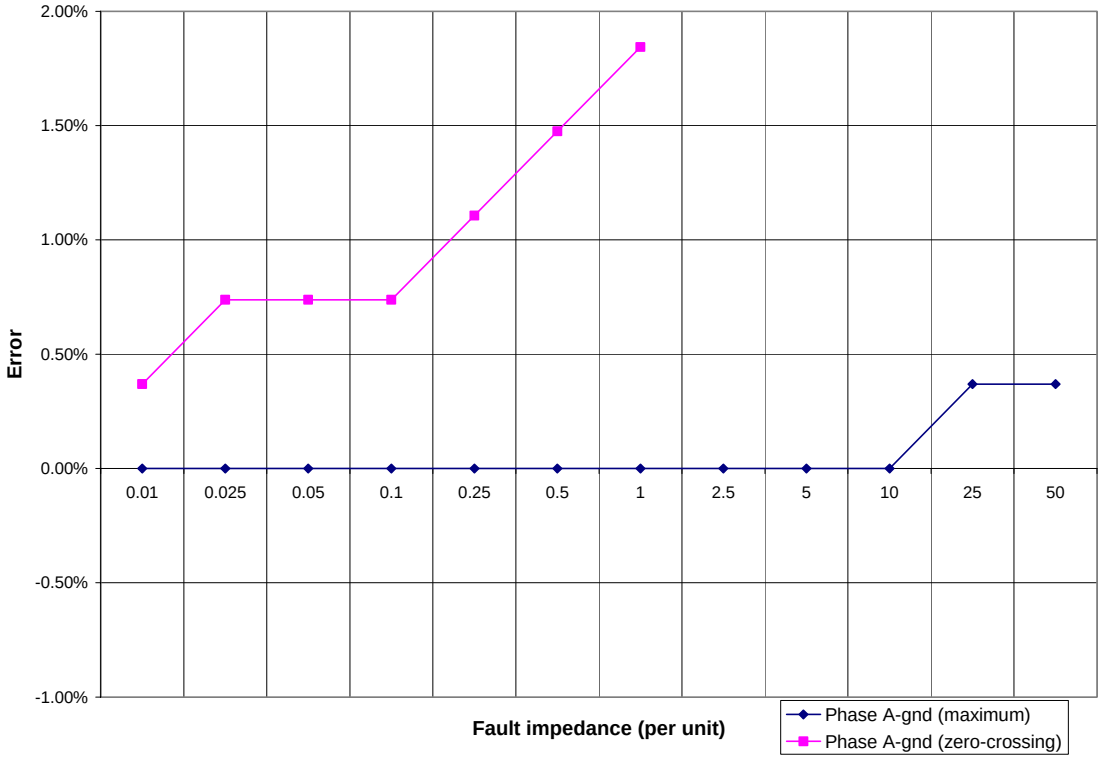


Figure 5-3 – Fault location error as a function of fault impedance for single line to ground faults

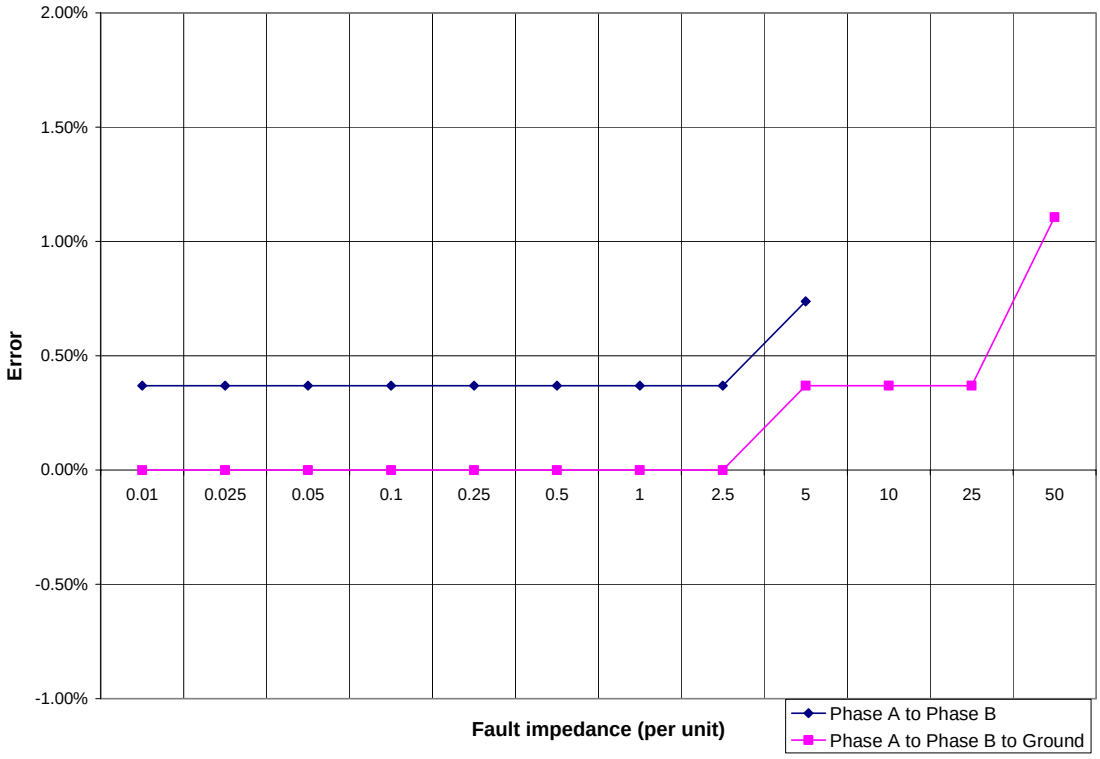


Figure 5-4 – Fault location error as a function of fault impedance for line to line faults

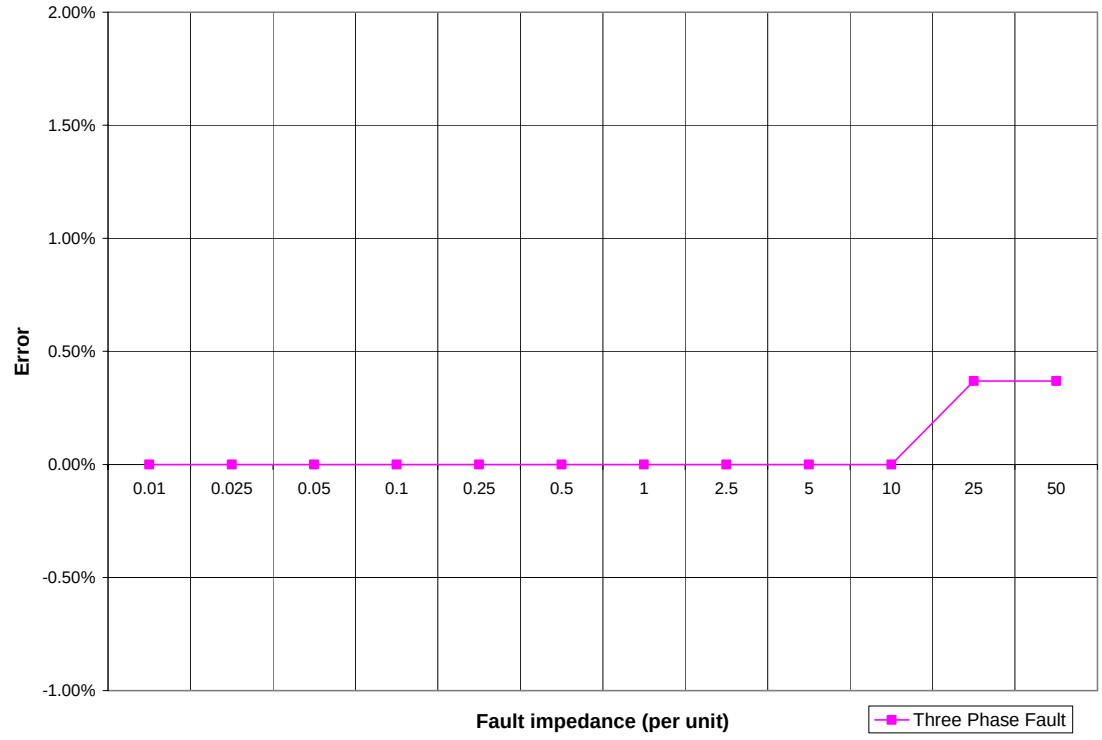


Figure 5-5 – Fault location error as a function of fault impedance for three phase faults

The slightly jagged nature of these Figures is due to the fact that the error is limited to a discrete set of values for a given fault location since the calculated fault locations are also limited to a discrete set of values. This due to the step size discussed in Section 3.5 (and specifically in Equation (64)) and is a result of the sampling rate. Higher sampling rates than the 2MHz which was used in these tests can decrease this appearance and may also improve the accuracy of detection, depending on the location of the fault in question.

The errors shown are on the order of hundreds of meters since 1% error is equal to 200m for the 20km long transmission line. This is in the range of the maximum error introduced by the synchronization using GPS [15]. As a result, significant improvements in the accuracy of these algorithms will not necessarily provide a worthwhile benefit at this time.

It is also important to note that the fault detection error and maximum detectable fault impedance are based on the margins of allowable operation for each algorithm. For example, the code in Appendix C which was used to carry out these tests requires the change in rho at any time to be greater than double the maximum detected change in rho for a fault to be detected. Reducing this allowable margin will increase the location accuracy for correct fault detections and will make higher impedances easier to locate but will also make it easier for the algorithm to detect faults incorrectly.

6. Conclusions

This thesis described the theory and methods of traveling wave fault detection and location using magnetic field sensing coils. The concept of the magnetic field for a general and three phase system was explored. This was followed by a presentation of the magnetic fields for a variety of conductor configurations and sensor locations. The four algorithms used in the magnetic field-based fault detection were then described. Finally, the combined algorithm was explained, and the results of accuracy and maximum detectable fault resistance were presented.

The magnetic field sensors were shown to be effective in detecting faults conceptually. Additionally, the collective algorithm was tested and was shown to provide accurate fault detection for relatively high fault impedances and for each common type of fault. All of this proves the magnetic field sensor to be a viable tool for power transmission line fault detection.

Future research could be performed in applying these algorithms to more complete systems than the single transmission line which was used for analysis in this thesis. Additionally, other fault location algorithms – most specifically, a fault location and classification scheme using the wavelet transform – could be modified to make use of the magnetic field. This will most likely improve the accuracy of fault location and increase the maximum detectable fault impedances.

Eventually a prototype of the magnetic field-based fault detector could be built and field tested. This would require more development of the sensor coils as well as harmonic-filtering circuitry. The MATLAB code would also need to be reconfigured since it is currently written to analyze pre-prepared sets of data to test the algorithm

rather than to continuously monitor information with which it is provided. The programming language would also most likely need to be changed to a different language which could be compiled directly for use with a microprocessor.

References

[1] T. Takagi, Y. Yamakoshi, M. Yamamura, R. Kondow, T. Matsushima, "Development of a New Type Fault Locator Using the One-Terminal Voltage and Current Data," in IEEE Transactions on Power Apparatus and Systems, Vol. PAS-101, No. 8, August 1982, pp. 2892-2898.

[2] D. A. Tziouvaras, J. B. Roberts, G. Benmouyal, "New Multi-Ended Fault Location Design for Two- or Three-Terminal Lines," in Developments in Power System Protection (IEE), Conference Publication No. 479, Amsterdam, 2001, pp. 395-398.

[3] IEEE Power Engineering Society (PES), IEEE Guide for Determining Fault Location on AC Transmission and Distribution Lines, IEEE Std. C37.114™-2004.

[4] K. Zimmerman, D. Costello, "Impedance-Based Fault Location Experience," in 2005 58th Annual Conference for Protective Relay Engineers, 2005, pp. 211-226.

[5] M. Sneddon, P. Gale, "Fault Location on Transmission Lines," in IEE Colloquium on Operational Monitoring of Distribution and Transmission Systems, January 1997, pp. 2/1-2/3.

[6] P. F. Gale, P. A. Crossley, X. Bingyin, G. Yaozhong, B. J. Cory, J. R. G. Barker, "Fault Location Based on Travelling Waves," in Fifth International Conference on Developments in Power System Protection, 1993, pp. 54-59.

[7] P. Crossley, "Future of the Global Positioning System in Power Systems," in IEE Colloquium on Developments in the Use of GPS in Power Systems, London, 8 February 1994, pp. 7/1-7/5.

[8] M. Aurangzeb, P. A. Crossley, P. Gale, "Fault Location on a Transmission Line Using High Frequency Travelling Waves Measured at a Single Line End," in Power Engineering Society Winter Meeting, Vol. 4, 2000, pp. 2437-2442.

[9] A. Elhaffar, M. Lehtonen, "Travelling Waves Based Earth Fault Location in 400kV Transmission Network Using Single End Measurement," in Large Engineering Systems Conference on Power Engineering, 2004, pp. 53-56.

[10] F. H. Magnago, A. Abur, "Fault Location Using Wavelets," in IEEE Transactions on Power Delivery, Vol. 13, No. 4, October 1998, p. 1475-1480.

[11] McBride. Fault Detector with Improved Response Time for Electrical Transmission System. Bridges Electric, Inc. Patent 4,408,155. 4 October 1983.

[12] Kejariwal, et al. Fault Detection and Location System for Power Transmission and Distribution Lines. The Research and Development Institute, Inc. at Montana State University. Patent 5,343,155. 30 August 1994.

[13] A. E. Emanuel, J. A. Orr, D. J. Pileggi, and E. M. Gulachenski, "A Non-Contact Technique for Determining Harmonic Currents Present in Individual Conductors of Overhead Lines," Presented at the IEEE PES 1982 Summer Meeting, San Francisco, July 1982.

[14] M. Vintan, "Fault Current Distribution Computation on Overhead Transmission Lines," in Proceedings of the Fifth International World Energy System Conference, vol. II, 2004, Oradea, Romania, pp. 273-279.

[15] J. Jiang, Y. Lin, J. Yang, T. Too, C. Liu, "An Adaptive PMU Based Fault Detection/Location Technique for Transmission Lines—Part II: PMU Implementation and Performance Evaluation," in IEEE Transactions on Power Delivery, Vol. 15, No. 4, October 2000, pp. 1136-1146.

[16] IEEE Power Systems Relaying Committee (PSRC), IEEE Guide for Protective Relay Applications to Transmission Lines, IEEE Std. C37.113-1999, pp. 31.

Appendix A : Model for Testing

The algorithm was tested for a 115kV transmission line using the power system simulation program ATPDraw as described in Section 5. The circuit for a ground fault simulation is shown below in Figure A-1; the data for the transmission line is shown in Figure A-2 and Figure A-3, and a diagram of the distances between the conductors and sensors is shown in Figure A-4.

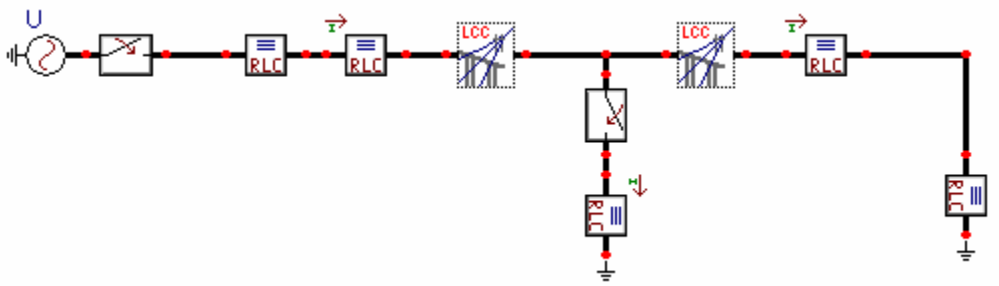


Figure A-1 – ATPDraw circuit for testing

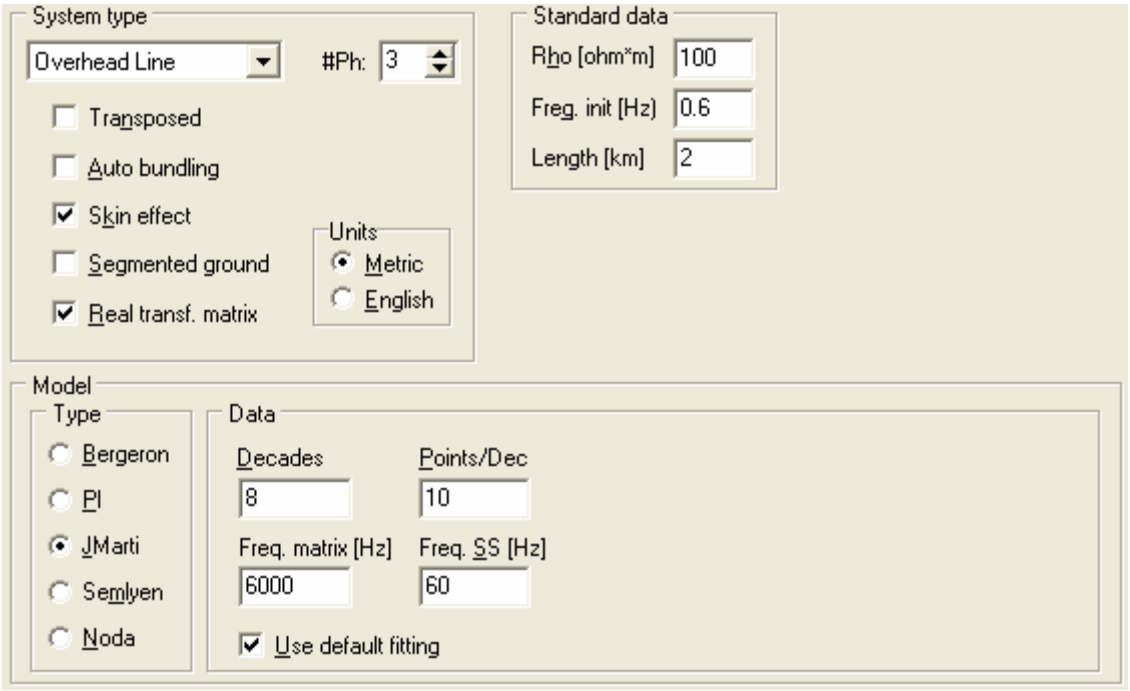


Figure A-2 – Modeling data for testing

	Ph.no.	Rin	Rout	Resis	Horiz	Vtower	Vmid
#		[cm]	[cm]	[ohm/km DC]	[m]	[m]	[m]
1	1	0.5	1.52	0.0614	-2	10	10
2	2	0.5	1.52	0.0614	0	10	10
3	3	0.5	1.52	0.0614	2	10	10
4	0	0	0.49	1.62	-1	11	11
5	0	0	0.49	1.62	1	11	11

Figure A-3 – Transmission line geometry for testing

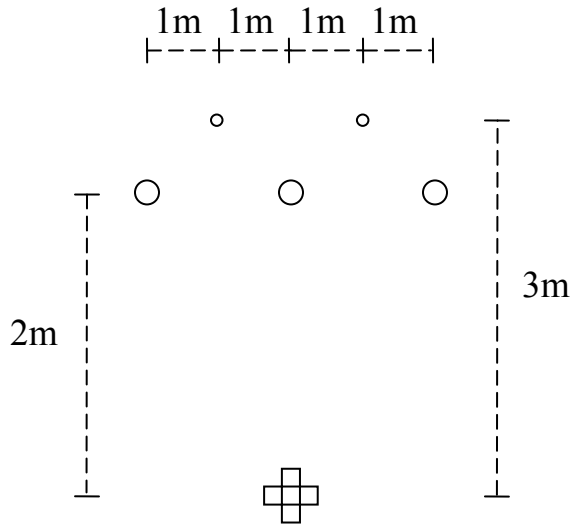


Figure A-4 – Conductor and sensor distance relationships for testing

Two identical transmission line models with different lengths connected in series were used in each test. A switch is connected to the node where these transmission lines meet; when the switch is opened, it connects a resistance to ground, thus simulating a ground fault. This resistance to ground is indicated by an RLC block. The lengths of each transmission line and the resistance to ground were changed for each test. The transmission line is terminated in a somewhat arbitrarily selected 500Ω resistive load (approximately 3.78 per unit, based on the per unit impedance of 132.25Ω found using Equation (68)) which is used to simulate a secondary-connected load; this is the RLC block on the far right. This allows an RMS current of approximately 132A to flow in

each phase during normal operation. The first RLC block is the generator impedance which was selected as approximately 0.17 per unit with an angle of approximately 80° since subtransient impedances are typically on this order of magnitude and are mostly reactive [1]. The other two RLC blocks which have not been described have a negligible resistance and are used to monitor the currents at the ends of the transmission line. These currents are later used in the analysis algorithm.

The transmission line model and geometry, shown in Figure A-2 and Figure A-3 respectively, are rather standard. The soil resistivity for this analysis was somewhat arbitrarily chosen as 100Ω·m since this is a typical soil resistivity for moist soils [2]. The skin effect option was used to allow ATP to generate the appropriate impedance for each conductor on its own.

The line model chosen was the JMarti model. This is a frequency-dependent transmission line model which uses a constant transformation matrix [3,4]. This is useful for fault detection studies since faults can cause high-frequency oscillations. As a result, a frequency-dependent model more accurately predicts the actual performance of a fault detection algorithm for a real transmission line.

The transmission line geometry chosen was a horizontal configuration. The dimensions are as shown in Figure A-3 and are based on Cardinal and Alumoweld (7 No. 8) conductors and the Horizontal Unshielded conductor configuration in the EPRI Transmission Line Reference Book [5]. The inner radii (used in calculating the skin effect) are of special note. The conductivity of steel is much lower than that of aluminum and thus the majority of current is located in the aluminum stranding of ACSR (aluminum conductor steel reinforced) conductors [6]. As a result, the steel stranding can

usually be neglected in calculations of impedance. The skin effect inner radius value allows ATP to take this into account in its calculation of line constants.

References for Appendix A

[1] Stevenson Jr., W. D., Elements of Power System Analysis, New York: McGraw-Hill, Inc., 1982.

[2] IEEE Power Engineering Society (PES), IEEE Guide for Safety in AC Substation Grounding, IEEE Std. 80-2000, pp. 50-53.

[3] L. Prikler, H. K. Høidalen, ATPDRAW Version 3.5 for Windows 9x/NT/2000/XP Users' Manual, Published through ATP-EMTP User Groups, October 2002, pp. 138, 140.

[4] Electro-Magnetic Transients Program (EMTP) Theory Book, Published through ATP-EMTP User Groups, July 1995, pp. 4-72–4-86.

[5] Electric Power Research Institute (EPRI), Transmission Line Reference Book on 115-138 kV Compact Line Design, EPRI Publication, 1983, pp. 5-11.

[6] G. Gaba and M. Abou-Dakka, "A Simplified and accurate calculation of Frequency Dependence Conductor Impedance," Presented at the 8th International Conference on Harmonics and Quality of Power (ICHQP), Athens, Greece, 1998.

Appendix B : Magnetic Field Plots by Fault Type

Some of these Figures have been presented previously. They are included both in the body of the thesis and in this Appendix for easy reference and for completeness. Note that each of these Figures is for a fault impedance of 1 per unit (which is equal to 132.25Ω as described in Section 5) and were tested using the circuit presented in Appendix A with each portion of the transmission line being 10km in length. The top charts for each fault type are at the generator end of the transmission line, while the bottom charts are for the load end. The first plot in each row is the full ellipse and beginning of the fault; the second plot is at a different scale so the details of the fault can be visually detected. The geometry is as described in Appendix A; it is reproduced here in Figure B-1.

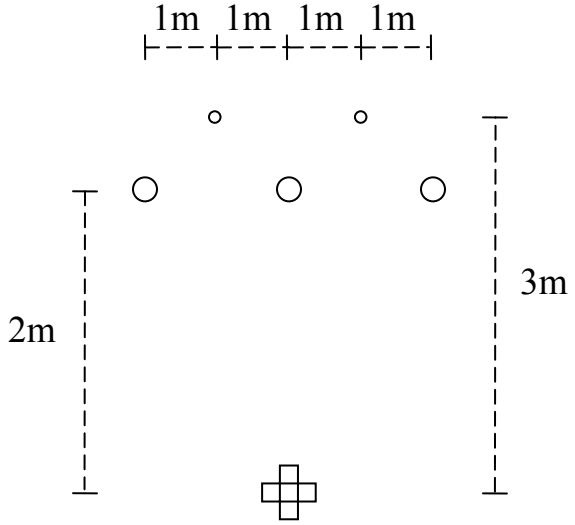


Figure B-1 – Conductor and sensor distance relationships for magnetic field fault plots

Line-to-Ground Fault: Phase a, at Faulted Current Peak

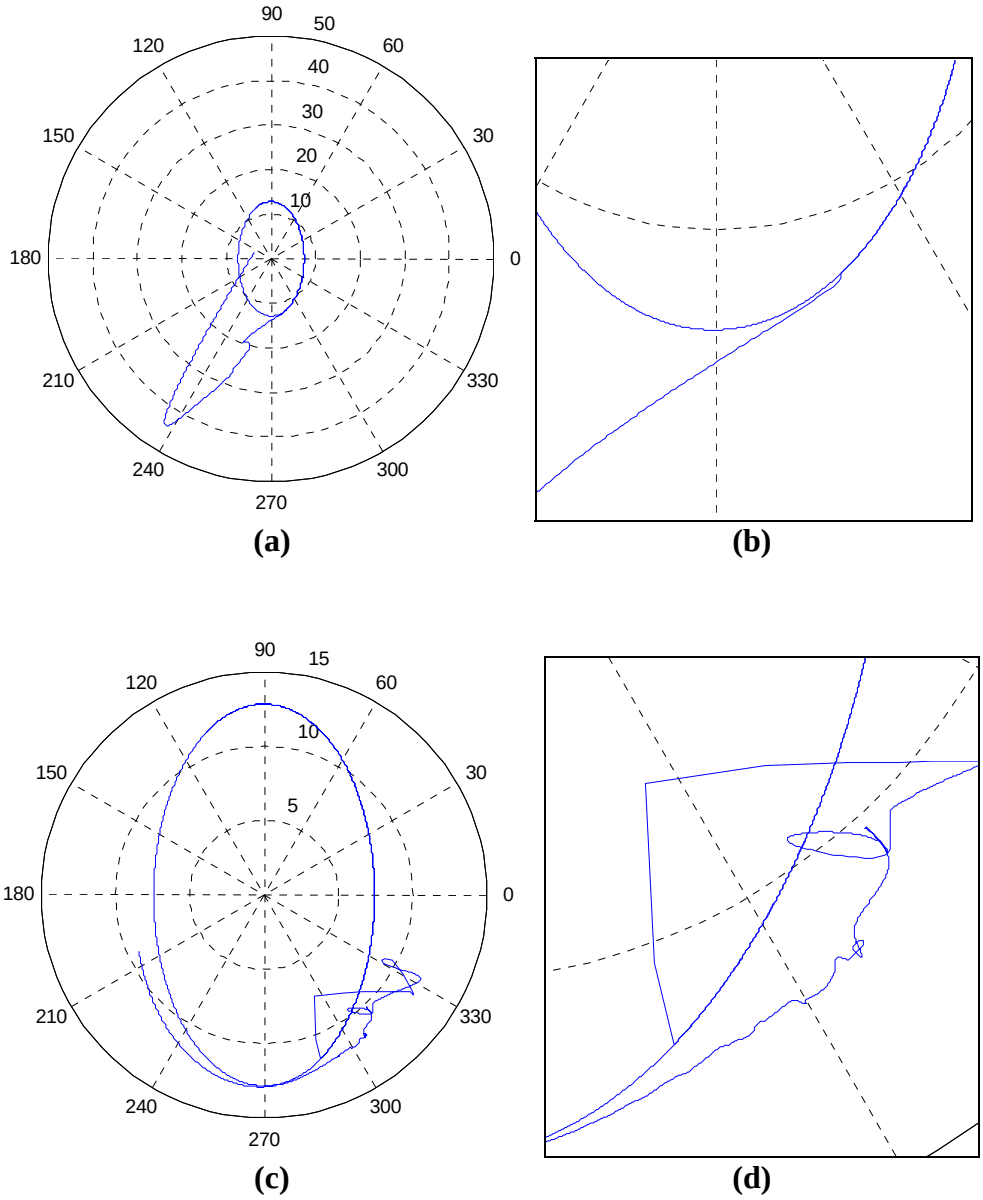


Figure B-2 – Line-to-Ground Fault: Phase a, fault connected at phase a current peak (a) – Generator end magnetic field; (b) – Generator end magnetic field fault detail; (c) – Load end magnetic field; (d) – Load end magnetic field fault detail

Line-to-Ground Fault: Phase a, at Faulted Current Zero

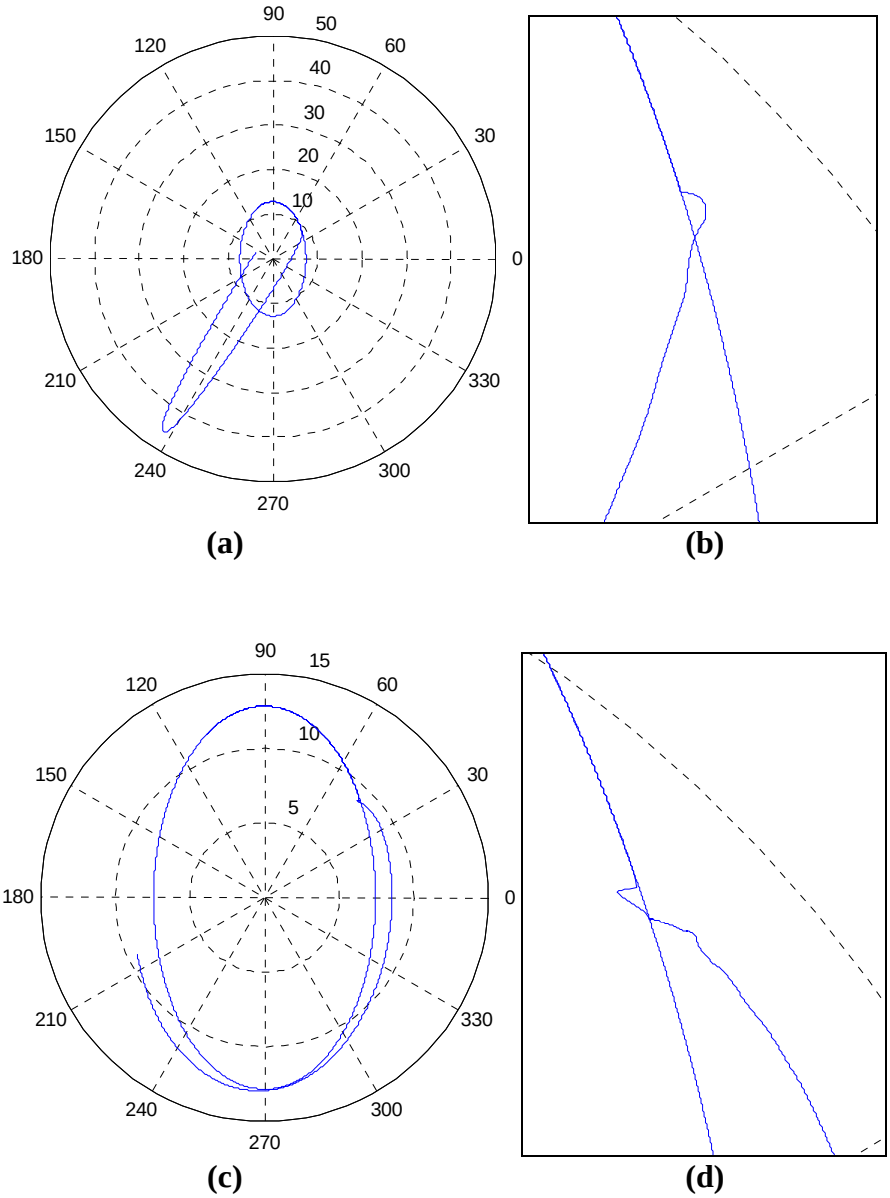


Figure B-3 – Line-to-Ground Fault: Phase a, fault connected at phase a current zero-crossing
(a) – Generator end magnetic field; (b) – Generator end magnetic field fault detail;
(c) – Load end magnetic field; (d) – Load end magnetic field fault detail

Line-to-Line Fault: Phases a and b

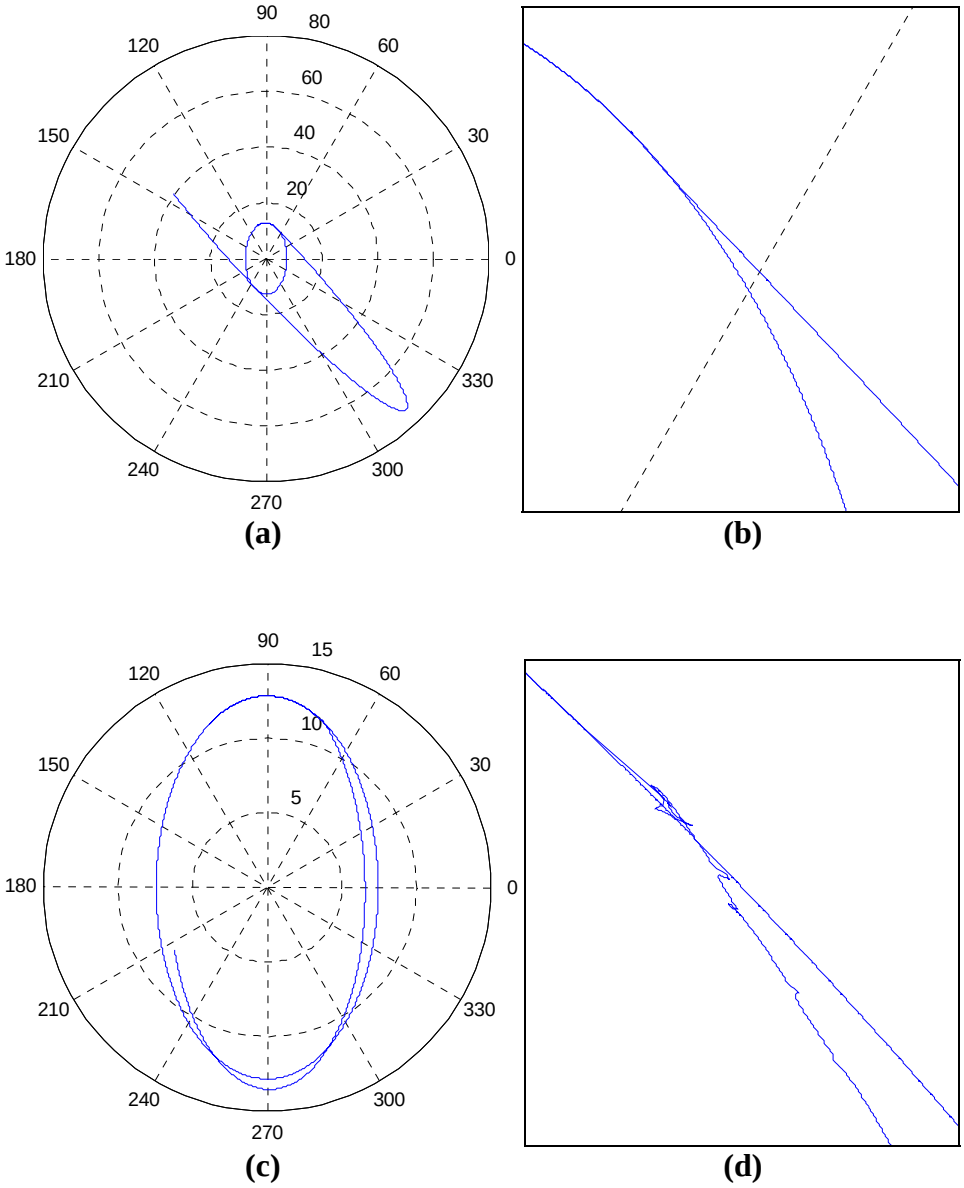


Figure B-4 – Line-to-Line Fault: Phases a and b
(a) – Generator end magnetic field; (b) – Generator end magnetic field fault detail;
(c) – Load end magnetic field; (d) – Load end magnetic field fault detail

Line-to-Line-to-Ground Fault: Phases a and b

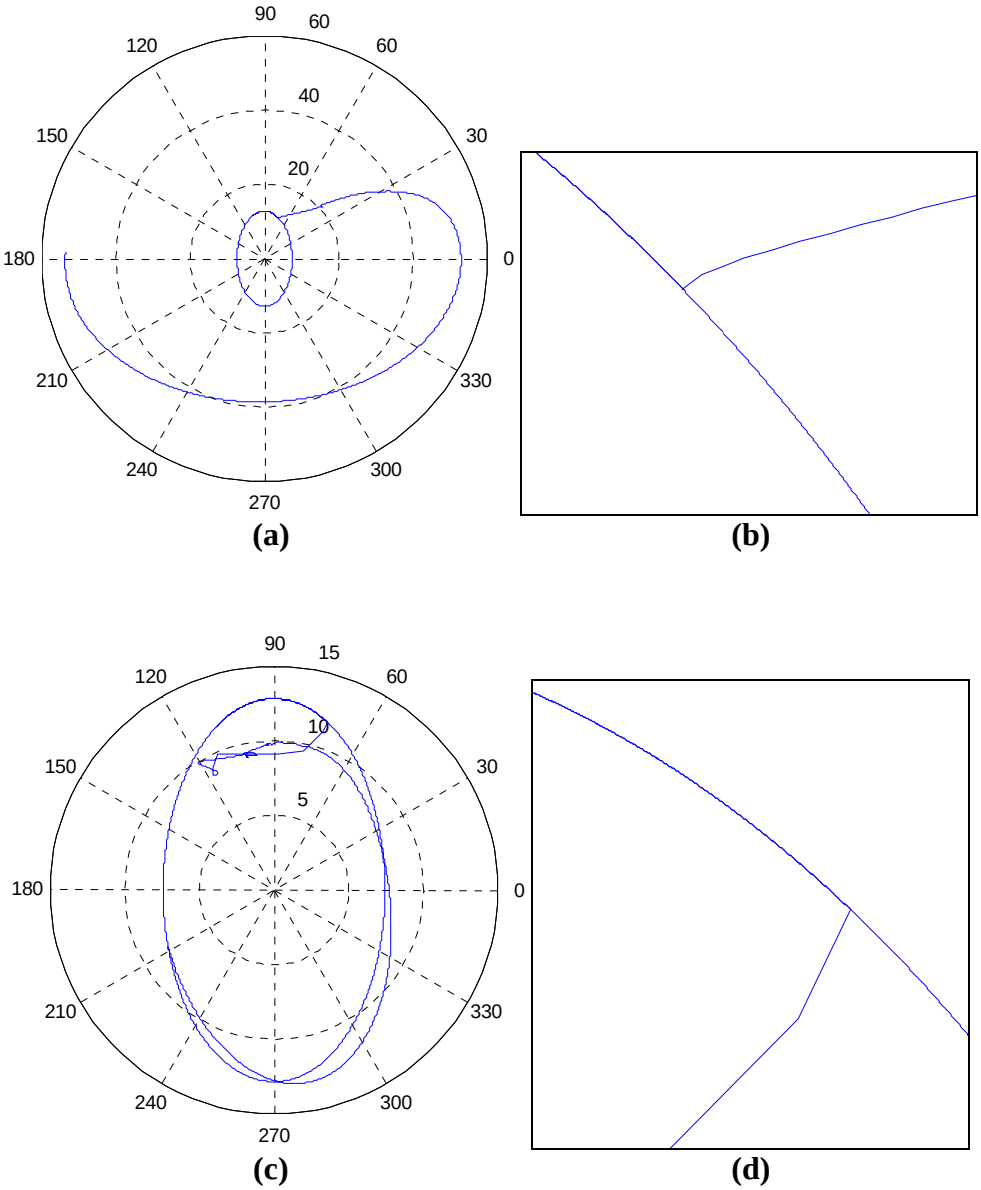


Figure B-5 – Line-to-Line-to-Ground Fault: Phases a and b to ground
(a) – Generator end magnetic field; (b) – Generator end magnetic field fault detail;
(c) – Load end magnetic field; (d) – Load end magnetic field fault detail

Three-Phase Fault

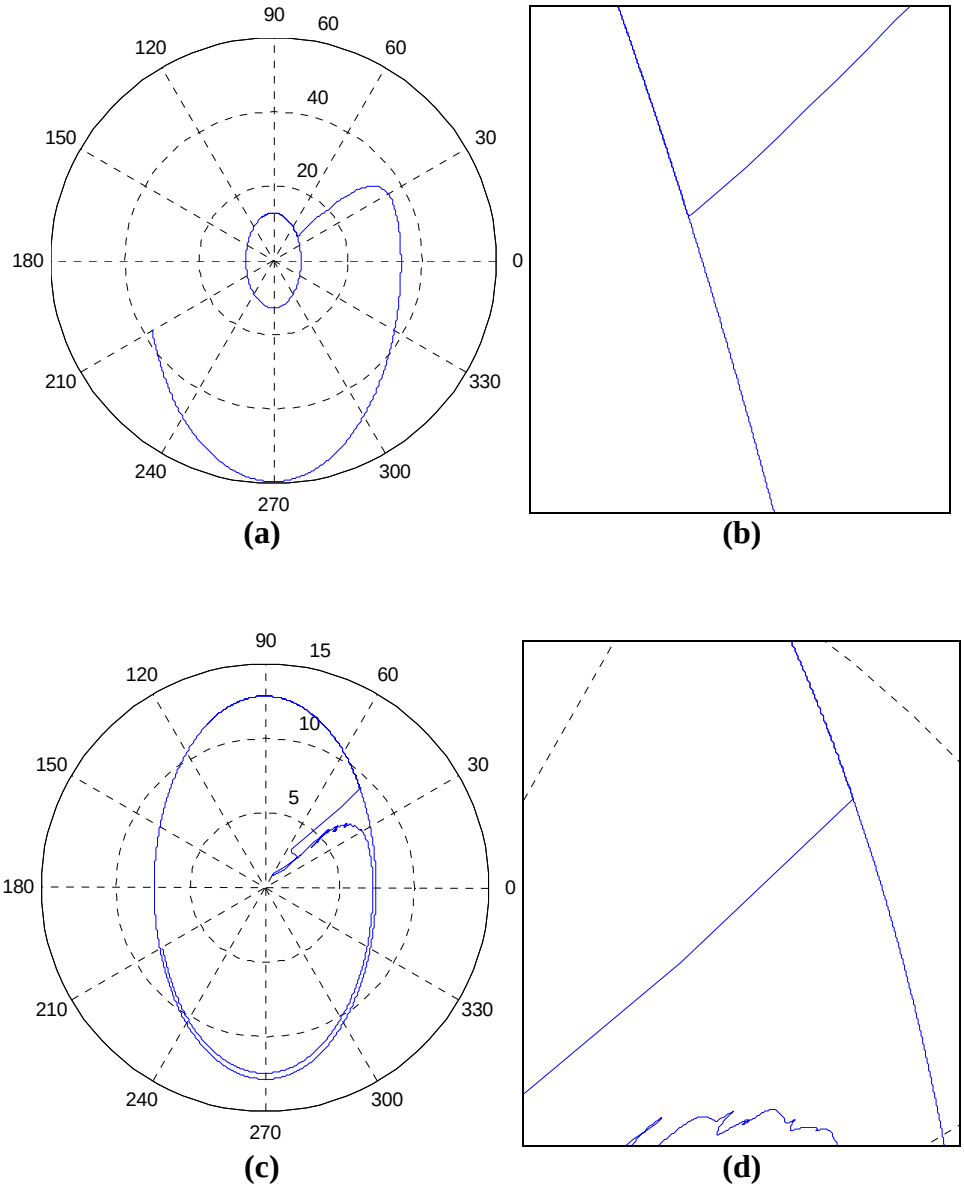


Figure B-6 – Three Phase Fault
(a) – Generator end magnetic field; (b) – Generator end magnetic field fault detail;
(c) – Load end magnetic field; (d) – Load end magnetic field fault detail

Appendix C : MATLAB Code

```
function [] =
hfieldxy(ia1,ib1,ic1,ia2,ib2,ic2,x,y1,y2,totlength,vel,tstart,tstep,bre
aktime)
%HFIELD Shows rotating field produced by transmission line currents.
%   Inputs are phase currents for phases a, b, and c, taken from an
%   Excel spreadsheet or the like.
%   Distance x is between conductors; distance y1 is between the
%   height of phases a and c and the height of b; y2 is between
%   the height of phases a and c and the height of the sensor.
%   totlength is the total length of the transmission line.
%   vel is the propagation velocity of the transmission line in m/s.
%   breaktime is the amonut of time to wait (in seconds) for a fault to
%   settle before resuming normal monitoring of the magnetic field
%   strengths.
%   Distances are in meters (except totlength, which is in km).
%   It is assumed that the sensor is directly underneath the b phase.
%
%   In detecting faults, the meanings of numbers attached
%   to error1recent and error2recent are as follows:
%   0 = no unusual behavior detected
%   1 = this sensor has detected a rise above the maximum allowable
%       value
%   2 = this sensor had detected a drop below the minimum allowable
%       value, then a rise above the minimum allowable value
%       approximately pi radians away from the point at which it
%       dropped (within a specified amount of time)
%   3 = this sensor has detected a rise above the "high" value
%   4 = this sensor has detected a drop below the minimum allowable
%       value without immediately returning above the minimum value
%
%   When there are two variables with almost identical names: variables
%   with "x" at the end are related to the position in an array (where
%   the unlabeled variables are related to actual time); variables with
%   "_p" are related to the previous value analysis; variables with
%   "_e" are used in the expected ellipse analysis; variables with "_d"
%   are used in the delta rho analysis; and variables with "_t" are
%   used in the delta theta analysis.
%
%   To read from an Excel spreadsheet:
%   ix = xlsread('filename.xls','Bx:By');



%Initial calculations and preparation for analysis

len = length(ia1);

rhoa = sqrt(x.^2 + y2.^2);     %Diagonal distances in conductor geometry.
rhob = abs(y2+y1);
rhoa2 = rhoa.^2;
```

```
Hx1 = 1./(2.*pi).*(ib1./rhob + ia1.*y2./rhoa2 + ic1.*y2./rhoa2);
Hy1 = 1./(2.*pi).*(ia1.*x./rhoa2 - ic1.*x./rhoa2);

Hx2 = 1./(2.*pi).*(ib2./rhob + ia2.*y2./rhoa2 + ic2.*y2./rhoa2);
Hy2 = 1./(2.*pi).*(ia2.*x./rhoa2 - ic2.*x./rhoa2);

%Adding noise
%Hx1 = Hx1 + wgn(len,1,-54,'dBm');
%Hy1 = Hy1 + wgn(len,1,-54,'dBm');
%Hx2 = Hx2 + wgn(len,1,-54,'dBm');
%Hy2 = Hy2 + wgn(len,1,-54,'dBm');


theta1 = atan2(Hy1,Hx1);    % converting time into theta and Hx
rho1 = (Hx1.^2 + Hy1.^2).^(1/2);  % and Hy into rho for polar plot
theta2 = atan2(Hy2,Hx2);
rho2 = (Hx2.^2 + Hy2.^2).^(1/2);


%Variable initialization


samplewaittime = (1./60).*(1./128); %In seconds - time to wait between
testing a rho and using it as the baseline for a test.
samplewaitx = round(samplewaittime./tstep.*10^6); %In samples
maxminwaittime = (1./60).*(1./2).*(0.9); %Minimum seconds to wait
between maxrhos and between minrhos (corresponds to 0.9 times half a
cycle - minimum usable frequency is 54Hz)
maxminwaitx = round(maxminwaittime./tstep.*10^6); %In samples

brkcorr = round(breaktime./tstep.*10^6);
maxtime = 0.001;                %Maximum time that the field can spend
below rhomin (in seconds) and still be classified as a fault.
maxtimex = round(maxtime./tstep.*10^6);
maxbelowtime = (1./60).*(1./8); %Maximum time to store data about a
drop below minrho.
maxbelowtimex = round(maxbelowtime./tstep.*10^6);
faultmaxtime = 1000.*totlength./vel; %Time to wait between seeing a
fault at one end of the line and the other end before determining that
the fault was erroneous.
faultmaxtimex = round(faultmaxtime.*1.5./tstep.*10^6);
emaxtime = (1./60).*(1./64);    %Min time between maximums and minimums.
emaxtimex = round(emaxtime./tstep.*10^6);
ewaittime = (1./60).*(1./16); %Difference in testing times for ellipse
maximum and minimum detection.
ewaittimex = round(ewaittime./tstep.*10^6);
quartertime = (1./60).*(1./4)*(19./20);
quartertimex = round(quartertime./tstep.*10^6);
eighthtime = (1./60).*(1./8);
eighthtimex = round(eighthtime./tstep.*10^6);
rhomaxtime = (1./60).*(1./256); %Time over which rhomax is averaged.
rhomaxtimex = round(rhomaxtime./tstep.*10^6);
thetamaxtime = (1./60).*(1./256); %Time over which thetamax is averaged.
thetamaxtimex = round(thetamaxtime./tstep.*10^6);
thetamintimex = thetamaxtimex;
```

```
misfaulttime = (1./60).*(1./4); %Min time between delta rho faults
before declaring misdetection.
misfaulttimex = round(misfaulttime./tstep.*10^6);
rhomaxrestoretime = 60*60;      %Time between changing rhomax and
restoring it.
rhomaxrestoretimex = round(rhomaxrestoretime./tstep.*10^6);
negdistance = 0;                %At end of code, indicates if a
negative distance is generated somehow.
fault = 0;
timestore = 0;                  %Time storage variable for analysis of
fault.

time = ewaittimex + 1;          %So time > 0 in all calculations.


fault_p = 0;
fault_e = 0;
fault_d = 0;
fault_t = 0;
error1recent_p = 0;
error2recent_p = 0;
error1recent_e = 0;
error2recent_e = 0;
error1recent_d = 0;
error2recent_d = 0;
error1recent_t = 0;
error2recent_t = 0;

newfault1time = [];
newfault2time = [];
fault1timecorr = [];
fault2timecorr = [];
timeneg = [];

deltheta1 = 0;
deltheta2 = 0;
thetatemp1a = 0;
thetatemp1b = 0;
thetatemp2a = 0;
thetatemp2b = 0;
delthetamax1 = 0;
delthetamax2 = 0;
delthetamin1 = 0;
delthetamin2 = 0;
delthetamax1dettime = 0;
delthetamax2dettime = 0;
delthetamin1dettime = 0;
delthetamin2dettime = 0;
deltheta1triptime = 0;
deltheta2triptime = 0;
faultwait_t = 0;
fault1time_t = [];
fault2time_t = [];
fault1timeonly_t = [];
fault2timeonly_t = [];
```

```
fault1timeonlyx_t = [];         %Separate storage for times when delta
algorithm detects
fault2timeonlyx_t = [];         %a fault independently
timeneg_t = [];

delrho1 = 0;
delrho2 = 0;
delrhomax1 = 10.^9;             %Maximum change in rho
delrhomax2 = 10.^9;
delrhomax1dettime = 0;
delrhomax2dettime = 0;
delrho1triptime = 0;
delrho2triptime = 0;
faultwait_d = 0;
misfaultcheck_d = 0;
misfaultwait_d = 0;
rhomaxrescount_d = 0;
tripchange_d = 0;
fault1time_d = [];
fault2time_d = [];
fault1timeonly_d = [];
fault2timeonly_d = [];
fault1timeonlyx_d = [];         %Separate storage for times when delta
algorithm detects
fault2timeonlyx_d = [];         %a fault independently
timeneg_d = [];

dropbelow1_p = 0;               %Indicates if the magnetic field has
dropped below the minimum rho value;
dropbelow2_p = 0;               %This is used in determining if the
load has simply decreased (or a fault has been cleared).
temptime1_p = 0;                %A temporary time value used in
analysis.
temptime2_p = 0;
temptheta1_p = 0;               %A temporary theta storage for the
possible fault.
temptheta2_p = 0;
faultwait_p = 0;                %Counting variable to test for
faultmaxtimex.
belowlowallow1_p = 0;
belowhighallow1_p = 0;
belowlowallow2_p = 0;
belowhighallow2_p = 0;
abovelowallow1_p = 0;
abovehighallow1_p = 0;
abovelowallow2_p = 0;
abovehighallow2_p = 0;
minrhoold1_p = 0;               %Minimum rhos from before a fault.
minrhoold2_p = 0;
fault1time_p = [];
fault2time_p = [];
fault1timeonly_p = [];          %Time at which only the previous value
fault2timeonly_p = [];          %algorithm finds a fault.
fault1type_p = [];
fault2type_p = [];
fault1typeonly_p = [];
```

```
fault2typeonly_p = [];
timeneg_p = [];
probtime1_p = [];
probtime2_p = [];

dropbelow1_e = 0;               %Indicates if the magnetic field has
dropped below the minimum rho value;
dropbelow2_e = 0;               %This is used in determining if the
load has simply decreased (or a fault has been cleared).
temptime1_e = 0;                %A temporary time value used in
analysis.
temptime2_e = 0;
temptheta1_e = 0;                %A temporary theta storage for the drop.
temptheta2_e = 0;
faultwait_e = 0;                %Counting variable to test for
faultmaxtimex.
minrho1_e = zeros(len,1);
minrho2_e = zeros(len,1);       %arbitrary large numbers, to prevent
maxrho1_e = 10^9.*ones(len,1);  %accidental fault detection before
maxrho2_e = 10^9.*ones(len,1);  %initialization these values have been
highrho1_e = 10^9.*ones(len,1); %defined
highrho2_e = 10^9.*ones(len,1);
mintheta_e = -10.*ones(len,1);
maxtheta_e = 10.*ones(len,1);
minrhoold1_e = zeros(len,1);    %minimum rhos affrom before a load
minrhoold2_e = zeros(len,1);
predrho1_e = zeros(len,1);
predrho2_e = zeros(len,1);
fault1time_e = [];
fault2time_e = [];
fault1timeonly_e = [];          %Time at which only the previous value
fault2timeonly_e = [];          %algorithm finds a fault.
fault1type_e = [];
fault2type_e = [];
fault1typeonly_e = [];
fault2typeonly_e = [];
timeneg_e = [];
probtime1_e = [];
probtime2_e = [];
lastrmintime1_e = 0;
lastrmintime2_e = 0;
lastrmaxtime1_e = 0;
lastrmaxtime2_e = 0;
rmin1_e = 0;
rmin2_e = 0;
rmax1_e = 10.^9;
rmax2_e = 10.^9;
wasgrowing1_e = 0;
wasgrowing2_e = 0;
thetashift1_e = 0;
thetashift2_e = 0;

lowallow1_p = 0.1;              %percentage below predicted allowed (1
= everything is allowed) before a time is logged
highallow1_p = 0.1;             %percentage above predicted allowed (1
= 100% above) before a time is logged
```

```
maxallow1_p = 0.3;              %percentage above predicted allowed (1
= 100% above) before indicating a fault
lowallow2_p = 0.1;
highallow2_p = 0.1;
maxallow2_p = 0.3;
lowtestlow1_p = 0.7.*lowallow1_p;   %"low" and "high" are reversed from
the logical expectation since both correspond
lowtesthigh1_p = 0.3.*lowallow1_p;  %to values subtracted (for the
lowtest variables)
lowtestlow2_p = 0.7.*lowallow2_p;
lowtesthigh2_p = 0.3.*lowallow2_p;
hightestlow1_p = 0.3.*highallow1_p;
hightesthigh1_p = 0.7.*highallow1_p;
hightestlow2_p = 0.3.*highallow2_p;
hightesthigh2_p = 0.7.*highallow2_p;

lowallow_e = 0.1;               %percentage below predicted allowed (1
= everything is allowed) before a time is logged
highallow_e = 0.1;              %percentage above predicted allowed (1
= 100% above) before a time is logged
maxallow_e = 0.3;               %percentage above predicted allowed (1
= 100% above) before indicating a fault

delrhoallow = 1;                %percentage above maximum change in rho
that is allowable
delrhoallowstart = delrhoallow; %reset value for delrhoallow

delthetaallowhigh = 0.75;        %percentage above maximum change in
theta
delthetaallowlow = 0.75;         %percentage below mimum change in
theta (1 = even no-change terms are allowable, negative = theta can
switch directions)

[wasgrowing1_e,wasgrowing2_e] = checkdir(rho1,rho2,time);
[rmin1_e,rmax1_e,rmin2_e,rmax2_e,thetashift1_e,thetashift2_e,wasgrowing
1_e,wasgrowing2_e,time,minrho1_e,highrho1_e,maxrho1_e,minrho2_e,highrho
2_e,maxrho2_e] =
minmaxrho(rho1,rho2,theta1,theta2,wasgrowing1_e,wasgrowing2_e,time,len,
lowallow_e,highallow_e,maxallow_e,ewaittimex,emaxtimex,quartertimex);

while time < samplewaitx + 1    %ensure that time > 0 in all
calculations
    time = time + 1;
end



%Main time loop for fault detection

while time < len


%Ensure no problems from previous iterations
```

```
    if (dropbelow1_p ~= 0)
        dropbelow1_p = dropbelow1_p + 1;     %Indicates that rho has
gone below the previous rhomin
    end
    if (dropbelow2_p ~= 0)
        dropbelow2_p = dropbelow2_p + 1;
    end
    if (dropbelow1_p == (maxbelowtimex+1))
        dropbelow1_p = 0;
    end
    if (dropbelow2_p == (maxbelowtimex+1))
        dropbelow2_p = 0;
    end

    if (dropbelow1_e ~= 0)
        dropbelow1_e = dropbelow1_e + 1;     %Indicates that rho has
gone below the previous rhomin
    end
    if (dropbelow2_e ~= 0)
        dropbelow2_e = dropbelow2_e + 1;
    end
    if (dropbelow1_e == (maxbelowtimex+1))
        dropbelow1_e = 0;
    end
    if (dropbelow2_e == (maxbelowtimex+1))
        dropbelow2_e = 0;
    end

    if (faultwait_p > faultmaxtimex)
        if (error1recent_p ~= 1) && (error2recent_p ~= 1)
            error1recent_p = 0;
            error2recent_p = 0;
            fault_p = 0;
        end
        if (error1recent_p == 1) || (error2recent_p == 1)
            if (error1recent_p == 1)
                probtime1_p =
cat(1,probtime1_p,temptime1_p.*tstep./10^6);
                temptime1_p = 0;
            end
            if (error2recent_p == 1)
                probtime2_p =
cat(1,probtime2_p,temptime2_p.*tstep./10^6);
                temptime2_p = 0;
            end
            error1recent_p = 0;
            error2recent_p = 0;
            fault_p = 0;
        end
        faultwait_p = 0;
    end
    if ((error1recent_p ~= 0) || (error2recent_p ~= 0))
        faultwait_p = faultwait_p + 1;
    end

    if (faultwait_e > faultmaxtimex)
```

```
        if (error1recent_e ~= 1) && (error2recent_e ~= 1)
            error1recent_e = 0;
            error2recent_e = 0;
            fault_e = 0;
        end
        if (error1recent_e == 1) || (error2recent_e == 1)
            if (error1recent_e == 1)
                probtime1_e =
cat(1,probtime1_e,temptime1_e.*tstep./10^6);
                temptime1_e = 0;
            end
            if (error2recent_e == 1)
                probtime2_e =
cat(1,probtime2_e,temptime2_e.*tstep./10^6);
                temptime2_e = 0;
            end
            error1recent_e = 0;
            error2recent_e = 0;
            fault_e = 0;
        end
        faultwait_e = 0;
    end
    if ((error1recent_e ~= 0) || (error2recent_e ~= 0))
        faultwait_e = faultwait_e + 1;
    end

    if (faultwait_d > faultmaxtimex)
        error1recent_d = 0;
        error2recent_d = 0;
        fault_d = 0;
        faultwait_d = 0;
    end
    if ((error1recent_d ~= 0) || (error2recent_d ~= 0))
        faultwait_d = faultwait_d + 1;
    end

    if (misfaultcheck_d >= 1)
        misfaultwait_d = misfaultwait_d + 1;
        if (misfaultwait_d > misfaulttimex)
            misfaultcheck_d = 0;
            misfaultwait_d = 0;
            tripchange_d = 0;
        end
    end
    if (misfaultcheck_d >= 3) && (tripchange_d == 0)
        delrhoallow = delrhoallow*2;
        tripchange_d = 1;
    end
    if (rhomaxrescount_d >= rhomaxrestoretimex)
        rhomaxrescount_d = 0;
        delrhoallow = delrhoallowstart;
        misfaultcheck_d = 0;
    end
    if (delrhoallow ~= delrhoallowstart)
        rhomaxrescount_d = rhomaxrescount_d + 1;
    end
```

```
    if (faultwait_t > faultmaxtimex)
        error1recent_t = 0;
        error2recent_t = 0;
        fault_t = 0;
        faultwait_t = 0;
    end
    if ((error1recent_t ~= 0) || (error2recent_t ~= 0))
        faultwait_t = faultwait_t + 1;
    end

%Initialize first variables for "delta rho" algorithm

    delrho1 = abs(rho1(time-1,1)-rho1(time,1));
    delrho2 = abs(rho2(time-1,1)-rho2(time,1));

%Initialize first variables for "delta theta" algorithm

    if ((theta1((time-1),1) > pi/2) && (theta1(time,1) < -
pi/2))    %this takes care of crossings at the boudary of the atan2
conversion (since theta is within [-pi,pi])
        thetatemp1a = theta1((time-1),1)-2*pi;
        thetatemp1b = theta1((time),1);
    elseif ((theta1((time-1),1) < -pi/2) && (theta1(time,1) > pi/2))
        thetatemp1a = theta1((time-1),1);
        thetatemp1b = theta1((time),1)-2*pi;
    else
        thetatemp1a = theta1((time-1),1);
        thetatemp1b = theta1((time),1);
    end
    if ((theta2((time-1),1) > pi/2) && (theta2(time,1) < -pi/2))
        thetatemp2a = theta2((time-1),1)-2*pi;
        thetatemp2b = theta2((time),1);
    elseif ((theta2((time-1),1) < -pi/2) && (theta2(time,1) > pi/2))
        thetatemp2a = theta2((time-1),1);
        thetatemp2b = theta2((time),1)-2*pi;
    else
        thetatemp2a = theta2((time-1),1);
        thetatemp2b = theta2((time),1);
    end

    deltheta1 = thetatemp1b-thetatemp1a;    %current theta minus
previous
    deltheta2 = thetatemp2b-thetatemp2a;



%Within main time loop - test for faults
%First, the "previous value" algorithm


    if (rho1(time,1) > (1 + maxallow1_p).*rho1(time-samplewaitx,1)) &&
(error1recent_p ~= 1)
        if (error1recent_p == 0)
            temptime1_p = time;
```

```
        end
        error1recent_p = 1;       %Indicate that a fault might have
occurred
        dropbelow1_p = 0;
        disp('Flag - Rise above max1p')  %test
    end
    if (rho2(time,1) > (1 + maxallow2_p).*rho2(time-samplewaitx,1)) &&
(error2recent_p ~= 1)
        if (error2recent_p == 0)
            temptime2_p = time;
        end
        error2recent_p = 1;       %indicate that a fault might have
occurred
        dropbelow2_p = 0;
        disp('Flag - Rise above max2p')  %test
    end

    if (rho1(time,1) < (1-lowallow1_p).*rho1(time-samplewaitx,1)) &&
(error1recent_p == 0) && (dropbelow1_p == 0) %tests for reduced load
accidentally tripping alarm
        dropbelow1_p = 1;
        minrhoold1_p = (1-lowallow1_p).*rho1(time-samplewaitx,1);
        temptime1_p = time;
        temptheta1_p = theta1(time,1);
        error1recent_p = 4;       %Indicate that a fault might have
occurred
        disp('Flag - Drop below min1p')  %test
    end
    if (rho2(time,1) < (1-lowallow2_p).*rho2(time-samplewaitx,1)) &&
(error2recent_p == 0) && (dropbelow2_p == 0)
        dropbelow2_p = 1;
        minrhoold2_p = (1-lowallow2_p).*rho2(time-samplewaitx,1);
        temptime2_p = time;
        temptheta2_p = theta2(time,1);
        error2recent_p = 4;       %Indicate that a fault might have
occurred
        disp('Flag - Drop below min2p')  %test
    end

    if (rho1(time,1) > (1 + highallow1_p).*rho1(time-samplewaitx,1)) &&
(error1recent_p == 0)
        if (dropbelow1_p ~= 0)        %Increases accuracy in the case
that the field crossed near the center
            temptime1_p = temptime1_p;  %by using the time that it
crossed the minimum rho value.
        else                          %In the case that this didn't
happen...
            temptime1_p = time;
        end
        error1recent_p = 3;       %Indicate that a fault might have
occurred
        disp('Flag - Rise above high1p')  %test
    end
    if (rho2(time,1) > (1 + highallow2_p).*rho2(time-samplewaitx,1)) &&
(error2recent_p == 0)
```

```
        if (dropbelow2_p ~= 0)        %Increases accuracy in the case
that the field crossed near the center
            temptime2_p = temptime2_p;  %by using the time that it
crossed the minimum rho value.
        else                          %In the case that this didn't
happen...
            temptime2_p = time;
        end
        error2recent_p = 3;       %indicate that a fault might have
occurred
        disp('Flag - Rise above high2p')  %test
    end

    if (rho1(time,1) > minrhoold1_p) && (dropbelow1_p ~= 0) &&
(abs(temptime1_p-time) < maxtimex) && (pi./2 < abs(theta1(time,1)-
temptheta1_p) < 3.*pi./2)
        temptime1_p = temptime1_p;
        dropbelow1_p = 0;
        error1recent_p = 2;       %indicate that a fault might have
occurred
        disp('Flag - Below-above fault 1p')  %test
    end
    if (rho2(time,1) > minrhoold2_p) && (dropbelow2_p ~= 0) &&
(abs(temptime2_p-time) < maxtimex) && (pi./2 < abs(theta2(time,1)-
temptheta2_p) < 3.*pi./2)
        temptime2_p = temptime2_p;
        dropbelow2_p = 0;
        error2recent_p = 2;       %indicate that a fault might have
occurred
        disp('Flag - Below-above fault 2p')  %test
    end


%Next, test with the "expected ellipse"

    if (rho1(time,1) > maxrho1_e(time,1)) && (error1recent_e ~= 1)
        if (error1recent_e == 0)
            temptime1_e = time;
        end
        error1recent_e = 1;       %Indicate that a fault might have
occurred
        dropbelow1_e = 0;
        disp('Flag - Rise above max1e')  %test
    end
    if (rho2(time,1) > maxrho2_e(time,1)) && (error2recent_e ~= 1)
        if (error2recent_e == 0)
            temptime2_e = time;
        end
        error2recent_e = 1;       %indicate that a fault might have
occurred
        dropbelow2_e = 0;
        disp('Flag - Rise above max2e')  %test
    end

    if (rho1(time,1) < minrho1_e(time,1)) && (error1recent_e == 0) &&
(dropbelow1_e == 0) %tests for reduced load accidentally tripping alarm
```

```
        dropbelow1_e = 1;
        minrhoold1_e = minrho1_e;
        temptime1_e = time;
        temptheta1_e = theta1(time,1);
        error1recent_e = 4;       %Indicate that a fault might have
occurred
        disp('Flag - Drop below min1e')  %test
    end
    if (rho2(time,1) < minrho2_e(time,1)) && (error2recent_e == 0) &&
(error2recent_e == 0) && (dropbelow2_e == 0)
        dropbelow2_e = 1;
        minrhoold2_e = minrho2_e;
        temptime2_e = time;
        temptheta2_e = theta2(time,1);
        error2recent_e = 4;       %Indicate that a fault might have
occurred
        disp('Flag - Drop below min2e')  %test
    end

    if (rho1(time,1) > highrho1_e(time,1)) && (error1recent_e == 0)
        if (dropbelow1_e ~= 0)        %Increases accuracy in the case
that the field crossed near the center
            temptime1_e = temptime1_e;  %by using the time that it
crossed the minimum rho value.
            dropbelow1_e = 0;
        else                          %In the case that this didn't
happen...
            temptime1_e = time;
        end
        error1recent_e = 3;       %Indicate that a fault might have
occurred
        disp('Flag - Rise above high1e')  %test
    end
    if (rho2(time,1) > highrho2_e(time,1)) && (error2recent_e == 0)
        if (dropbelow2_e ~= 0)        %Increases accuracy in the case
that the field crossed near the center
            temptime2_e = temptime2_e;  %by using the time that it
crossed the minimum rho value.
            dropbelow2_e = 0;
        else                          %In the case that this didn't
happen...
            temptime2_e = time;
        end
        error2recent_e = 3;       %indicate that a fault might have
occurred
        disp('Flag - Rise above high2e')  %test
    end

    if (rho1(time,1) > minrhoold1_e(time,1)) && (dropbelow1_e ~= 0) &&
(abs(temptime1_e-time) < maxtimex) && (pi./2 < abs(theta1(time,1)-
temptheta1_e) < 3.*pi./2)
        temptime1_e = temptime1_e
        dropbelow1_e = 0;
        error1recent_e = 2;       %indicate that a fault might have
occurred
        disp('Flag - Below-above fault 1e')  %test
```

```
    end
    if (rho2(time,1) > minrhoold2_e(time,1)) && (dropbelow2_e ~= 0) &&
(abs(temptime2_e-time) < maxtimex) && (pi./2 < abs(theta2(time,1)-
temptheta2_e) < 3.*pi./2)
        temptime2_e = temptime2_e
        dropbelow2_e = 0;
        error2recent_e = 2;       %indicate that a fault might have
occurred
        disp('Flag - Below-above fault 2e')  %test
    end

%Now with "delta rho" algorithm

    if (delrho1 > delrhomax1.*(1+delrhoallow)) && (error1recent_d == 0)
        delrho1triptime = time
        error1recent_d = 1;
        misfaultcheck_d = misfaultcheck_d + 1;
    end
    if (delrho2 > delrhomax2.*(1+delrhoallow)) && (error2recent_d == 0)
        delrho2triptime = time
        error2recent_d = 1;
        misfaultcheck_d = misfaultcheck_d + 1;
    end

%Now with "delta theta" algorithm

    if (delthetamax1 >= 0) && (delthetamin1 >= 0)
        if (delthetamax1 ~= 0) && ((deltheta1 >
delthetamax1.*(1+delthetaallowhigh)) || (deltheta1 < delthetamin1.*(1-
delthetaallowlow)))&& (error1recent_t == 0)
            deltheta1triptime = time
            error1recent_t = 1;
        end
    else
        if (delthetamax1 ~= 0) && ((deltheta1 <
delthetamax1.*(1+delthetaallowhigh)) || (deltheta1 > delthetamin1.*(1-
delthetaallowlow)))&& (error1recent_t == 0)
            deltheta1triptime = time
            error1recent_t = 1;
            delthetamax1
            delthetamin1
            deltheta1
        end
    end
    if (delthetamax2 >= 0) && (delthetamin2 >= 0)
        if (delthetamax2 ~= 0) && ((deltheta2 >
delthetamax2.*(1+delthetaallowhigh)) || (deltheta1 < delthetamin2.*(1-
delthetaallowlow)))&& (error2recent_t == 0)
            deltheta2triptime = time
            error2recent_t = 1;
        end
    else
        if (delthetamax2 ~= 0) && ((deltheta2 <
delthetamax2.*(1+delthetaallowhigh)) || (deltheta1 > delthetamin2.*(1-
delthetaallowlow)))&& (error2recent_t == 0)
            deltheta2triptime = time
```

```
            error2recent_t = 1;
        end
    end



%Variable re-initialization
%Initialize variables for "expected ellipse" algorithm

    if (rho1((time-ewaittimex),1) > rho1(time,1)) && (wasgrowing1_e==1)
&& (time > (lastrmintime1_e + emaxtimex))
        timemaxcheck = 0;
        rmax1_e = 0;
        while (timemaxcheck < quartertimex) && ((time-timemaxcheck) >
0)
            if (rho1((time-timemaxcheck),1) > rmax1_e)
                rmax1_e = rho1((time-timemaxcheck),1);
                thetashift1_e = theta1((time-timemaxcheck),1);
            end
            timemaxcheck = timemaxcheck + 1;
        end
        lastrmaxtime1_e = time;
        wasgrowing1_e = 0;
    end
    if (rho2((time-ewaittimex),1) > rho2(time,1)) && (wasgrowing2_e==1)
&& (time > (lastrmintime2_e + emaxtimex))
        timemaxcheck = 0;
        rmax2_e = 0;
        while (timemaxcheck < quartertimex) && ((time-timemaxcheck) >
0)
            if (rho2((time-timemaxcheck),1) > rmax2_e)
                rmax2_e = rho2((time-timemaxcheck),1);
                thetashift2_e = theta2((time-timemaxcheck),1);
            end
            timemaxcheck = timemaxcheck + 1;
        end
        lastrmaxtime2_e = time;
        wasgrowing2_e = 0;
    end
    if (rho1((time-ewaittimex),1) < rho1(time,1)) && (wasgrowing1_e==0)
&& (time > (lastrmaxtime1_e + emaxtimex))
        timemincheck = 0;
        rmin1_e = 10.^9;
        while (timemincheck < quartertimex) && ((time-timemincheck) >
0)
            if (rho1((time-timemincheck),1) < rmin1_e)
                rmin1_e = rho1((time-timemincheck),1);
            end
            timemincheck = timemincheck + 1;
        end
        lastrmintime1_e = time;
        wasgrowing1_e = 1;
        if rmin1_e < 0.1
            rmin1_e = 0.1;
        end
    end
```

```
    if (rho2((time-ewaittimex),1) < rho2(time,1)) && (wasgrowing2_e==0)
&& (time > (lastrmaxtime2_e + emaxtimex))
        timemincheck = 0;
        rmin2_e = 10.^9;
        while (timemincheck < quartertimex) && ((time-timemincheck) >
0)
            if (rho2((time-timemincheck),1) < rmin2_e)
                rmin2_e = rho2((time-timemincheck),1);
            end
            timemincheck = timemincheck + 1;
        end
        lastrmintime2_e = time;
        wasgrowing2_e = 1;
        if rmin2_e < 0.1
            rmin2_e = 0.1;
        end

        predrho1_e = (rmax1_e.*rmin1_e)./((rmax1_e.*sin(theta1-
thetashift1_e)).^2+(rmin1_e.*cos(theta1-
thetashift1_e)).^2+0.0001).^(1/2);
        predrho2_e = (rmax2_e.*rmin2_e)./((rmax2_e.*sin(theta2-
thetashift2_e)).^2+(rmin2_e.*cos(theta2-
thetashift2_e)).^2+0.0001).^(1/2);

        if (rmin1_e == 0.1)
            minrho1_e = zeros(len,1);
        else
            minrho1_e = predrho1_e.*(1-lowallow_e);
        end
        if (rmin2_e == 0.1)
            minrho2_e = zeros(len,1);
        else
            minrho2_e = predrho2_e.*(1-lowallow_e);
        end
        highrho1_e = predrho1_e.*(1+highallow_e);
        highrho2_e = predrho2_e.*(1+highallow_e);
        maxrho1_e = predrho1_e.*(1+maxallow_e);
        maxrho2_e = predrho2_e.*(1+maxallow_e);
    end

%Initialize more "delta rho" algorithm variables

    if (time > (delrhomax1dettime + quartertimex)) &&
((((theta1(time,1) > thetashift1_e + pi./4 - pi./128) &&
(theta1(time,1) < thetashift1_e + pi./4 + pi./128)) || ((theta1(time,1)
> thetashift1_e + 3.*pi./4 - pi./128) && (theta1(time,1) <
thetashift1_e + 3.*pi./4 + pi./128)) || ((theta1(time,1) >
thetashift1_e - pi./4 - pi./128) && (theta1(time,1) < thetashift1_e -
pi./4 + pi./128)) || ((theta1(time,1) > thetashift1_e - 3.*pi./4 -
pi./128) && (theta1(time,1) < thetashift1_e - 3.*pi./4 +
pi./128))) )%%|| ((time >= eighthtimex + lastrmaxtime1_e) && (time >=
eighthtimex + lastrmintime1_e)))
        timerhotest = 0;
        delrholist = [];
        while (timerhotest < rhomaxtimex)
```

```
            delrholist = cat(1,delrholist,abs(rho1(time-
(timerhotest),1)-rho1(time-(timerhotest+1),1)));
            timerhotest = timerhotest + 1;
        end
        delrhomax1 = mean(delrholist);
        delrhomax1dettime = time;
    end
    if (time > (delrhomax2dettime + quartertimex)) &&
((((theta2(time,1) > thetashift2_e + pi./4 - pi./128) &&
(theta2(time,1) < thetashift2_e + pi./4 + pi./128)) || ((theta2(time,1)
> thetashift2_e + 3.*pi./4 - pi./128) && (theta2(time,1) <
thetashift2_e + 3.*pi./4 + pi./128)) || ((theta2(time,1) >
thetashift2_e - pi./4 - pi./128) && (theta2(time,1) < thetashift2_e -
pi./4 + pi./128)) || ((theta2(time,1) > thetashift2_e - 3.*pi./4 -
pi./128) && (theta2(time,1) < thetashift2_e - 3.*pi./4 +
pi./128))) )%%|| ((time >= eighthtimex + lastrmaxtime2_e) && (time >=
eighthtimex + lastrmintime2_e)))
        timerhotest = 0;
        delrholist = [];
        while (timerhotest < rhomaxtimex)
            delrholist = cat(1,delrholist,abs(rho2(time-
(timerhotest),1)-rho2(time-(timerhotest+1),1)));
            timerhotest = timerhotest + 1;
        end
        delrhomax2 = mean(delrholist);
        delrhomax2dettime  = time;
    end

%Initialize more "delta theta" algorithm variables

    if (time == lastrmintime1_e) && (time > (delthetamax1dettime +
quartertimex))
        timethetatest = 0;
        delthetalist = [];
        while (timethetatest < thetamaxtimex)
            delthetalist = cat(1,delthetalist,(theta1(time-
(timethetatest),1)-theta1(time-(timethetatest+1),1)));
            timethetatest = timethetatest + 1;
        end
        delthetamax1 = mean(delthetalist);
        delthetamax1dettime = time;
    end
    if (time == lastrmintime2_e) && (time > (delthetamax2dettime +
quartertimex))
        timethetatest = 0;
        delthetalist = [];
        while (timethetatest < thetamaxtimex)
            delthetalist = cat(1,delthetalist,(theta2(time-
(timethetatest),1)-theta2(time-(timethetatest+1),1)));
            timethetatest = timethetatest + 1;
        end
        delthetamax2 = mean(delthetalist);
        delthetamax2dettime = time;
    end
    if (time == lastrmaxtime1_e) && (time > (delthetamin1dettime +
quartertimex))
```

```
        timethetatest = 0;
        delthetalist = [];
        while (timethetatest < thetamintimex)
            delthetalist = cat(1,delthetalist,(theta1(time-
(timethetatest),1)-theta1(time-(timethetatest+1),1)));
            timethetatest = timethetatest + 1;
        end
        delthetamin1 = mean(delthetalist);
        delthetamin1dettime = time;
    end
    if (time == lastrmaxtime2_e) && (time > (delthetamin2dettime +
quartertimex))
        timethetatest = 0;
        delthetalist = [];
        while (timethetatest < thetamintimex)
            delthetalist = cat(1,delthetalist,(theta2(time-
(timethetatest),1)-theta2(time-(timethetatest+1),1)));
            timethetatest = timethetatest + 1;
        end
        delthetamin2 = mean(delthetalist);
        delthetamin2dettime = time;
    end

%Adjust minallow, highallow, and maxallow for "previous value"
algorithm

    if (rho1(time,1) < (1 - lowtestlow1_p).*rho1(time-samplewaitx,1))
&& (error1recent_p == 0)
        belowlowallow1_p = 1;
    end
    if (rho1(time,1) < (1 - lowtesthigh1_p).*rho1(time-samplewaitx,1))
&& (error1recent_p == 0)
        abovelowallow1_p = 1;
    end
    if (rho1(time,1) > (1 + hightestlow1_p).*rho1(time-samplewaitx,1))
&& (error1recent_p == 0)
        belowhighallow1_p = 1;
    end
    if (rho1(time,1) > (1 + hightesthigh1_p).*rho1(time-samplewaitx,1))
&& (error1recent_p == 0)
        abovehighallow1_p = 1;
    end

    if ((theta1(time-1,1) >= 0) && (theta1(time,1) <= 0 )) ||
((theta1(time,1) >= 0) && (theta1(time-1,1) <= 0 ))
        if belowlowallow1_p == 1                %If we've gone below
the lower adjustment threshold
            lowallow1_p = lowallow1_p + 0.05;
            lowtestlow1_p = 0.7.*lowallow1_p;
            lowtesthigh1_p = 0.3.*lowallow1_p;
        end
        if abovelowallow1_p ~= 1                %If we haven't gone
below the higher adjustment threshold
            lowallow1_p = lowallow1_p - 0.01;
            lowtestlow1_p = 0.7.*lowallow1_p;
            lowtesthigh1_p = 0.3.*lowallow1_p;
```

```
        end
        if belowhighallow1_p ~= 1
            highallow1_p = highallow1_p - 0.01;
            maxallow1_p = maxallow1_p - 0.02;
            hightestlow1_p = 0.3.*highallow1_p;
            hightesthigh1_p = 0.7.*highallow1_p;
        end
        if abovehighallow1_p == 1
            highallow1_p = highallow1_p + 0.05;
            maxallow1_p = maxallow1_p + 0.1;
            hightestlow1_p = 0.3.*highallow1_p;
            hightesthigh1_p = 0.7.*highallow1_p;
        end
        belowlowallow1_p = 0;
        abovelowallow1_p = 0;
        belowhighallow1_p = 0;
        abovehighallow1_p = 0;
    end

    if (rho2(time,1) < (1 - lowtestlow2_p).*rho2(time-samplewaitx,1))
&& (error2recent_p == 0)
        belowlowallow2_p = 1;
    end
    if (rho2(time,1) < (1 - lowtesthigh2_p).*rho2(time-samplewaitx,1))
&& (error2recent_p == 0)
        abovelowallow2_p = 1;
    end
    if (rho2(time,1) > (1 + hightestlow2_p).*rho2(time-samplewaitx,1))
&& (error2recent_p == 0)
        belowhighallow2_p = 1;
    end
    if (rho2(time,1) > (1 + hightesthigh2_p).*rho2(time-samplewaitx,1))
&& (error2recent_p == 0)
        abovehighallow2_p = 1;
    end

    if ((theta2(time-1,1) >= 0) && (theta2(time,1) <= 0 )) ||
((theta2(time,1) >= 0) && (theta2(time-1,1) <= 0 ))
        if belowlowallow2_p == 1                %If we've gone below
the lower adjustment threshold
            lowallow2_p = lowallow2_p + 0.05;
            lowtestlow2_p = 0.7.*lowallow2_p;
            lowtesthigh2_p = 0.3.*lowallow2_p;
        end
        if abovelowallow2_p ~= 1                %If we haven't gone
below the higher adjustment threshold
            lowallow2_p = lowallow2_p - 0.01;
            lowtestlow2_p = 0.7.*lowallow2_p;
            lowtesthigh2_p = 0.3.*lowallow2_p;
        end
        if belowhighallow2_p ~= 1
            highallow2_p = highallow2_p - 0.01;
            maxallow2_p = maxallow2_p - 0.02;
            hightestlow2_p = 0.3.*highallow2_p;
            hightesthigh2_p = 0.7.*highallow2_p;
        end
```

```
        if abovehighallow2_p == 1
            highallow2_p = highallow2_p + 0.05;
            maxallow2_p = maxallow2_p + 0.1;
            hightestlow2_p = 0.3.*highallow2_p;
            hightesthigh2_p = 0.7.*highallow2_p;
        end
        belowlowallow2_p = 0;
        abovelowallow2_p = 0;
        belowhighallow2_p = 0;
        abovehighallow2_p = 0;
    end




%Determining faults

    if ((error1recent_e ~= 0) && (error2recent_e ~= 0) && (fault_e ==
0))   %Something has happened at both ends
        disp('Flag - Fault E')  %test
        fault_e = 1;
        fault1time_e = cat(1,fault1time_e,temptime1_e);
        fault1type_e = cat(1,fault1type_e,error1recent_e);
        fault2time_e = cat(1,fault2time_e,temptime2_e);
        fault2type_e = cat(1,fault2type_e,error2recent_e);
    end

    if ((error1recent_p ~= 0) && (error2recent_p ~= 0) && (fault_p ==
0))   %Something has happened at both ends
        disp('Flag - Fault P')  %test
        fault_p = 1;
        fault1time_p = cat(1,fault1time_p,temptime1_p);
        fault1type_p = cat(1,fault1type_p,error1recent_p);
        fault2time_p = cat(1,fault2time_p,temptime2_p);
        fault2type_p = cat(1,fault2type_p,error2recent_p);
    end

    if ((error1recent_d ~= 0) && (error2recent_d ~= 0) && (fault_d ==
0))   %Something has happened at both ends
        disp('Flag - Fault D')  %test
        fault = 1;
        fault_d = 1;
        fault1timeonlyx_d = cat(1,fault1timeonlyx_d,delrho1triptime);
        fault2timeonlyx_d = cat(1,fault2timeonlyx_d,delrho2triptime);
        delrhomax1 = 10.^9;      %prevent repetitive tripping
        delrhomax2 = 10.^9;
    end

    if ((error1recent_t ~= 0) && (error2recent_t ~= 0) && (fault_t ==
0))   %Something has happened at both ends
        disp('Flag - Fault T')  %test
        fault = 1;
        fault_t = 1;
        fault1timeonlyx_t = cat(1,fault1timeonlyx_t,deltheta1triptime);
        fault2timeonlyx_t = cat(1,fault2timeonlyx_t,deltheta2triptime);
```

```
        delthetamax1 = 0;      %prevent repetitive tripping
        delthetamax2 = 0;
        delthetamin1 = 0;
        delthetamin2 = 0;
    end

    if ((faultwait_e >= faultmaxtimex) && fault_e) || ((faultwait_p >=
faultmaxtimex) && fault_p) || (fault_e && fault_p) || (fault_d &&
fault_t)
        disp('Flag - Fault Both - or over wait')  %test
        fault = 1;
        if ~fault_p
            fault1time_p = cat(1,fault1time_p,'X');
            fault2time_p = cat(1,fault2time_p,'X');
            fault1type_p = cat(1,fault1type_p,'X');
            fault2type_p = cat(1,fault2type_p,'X');
        end
        if ~fault_e
            fault1time_e = cat(1,fault1time_e,'X');
            fault2time_e = cat(1,fault2time_e,'X');
            fault1type_e = cat(1,fault1type_e,'X');
            fault2type_e = cat(1,fault2type_e,'X');
        end
        if fault_d
            fault1time_d =
cat(1,fault1time_d,fault1timeonlyx_d(length(fault1timeonlyx_d),1));
            fault2time_d =
cat(1,fault2time_d,fault2timeonlyx_d(length(fault2timeonlyx_d),1));
            fault1timeonlyx_d =
removerows(fault1timeonlyx_d,length(fault1timeonlyx_d));
            fault2timeonlyx_d =
removerows(fault2timeonlyx_d,length(fault2timeonlyx_d));
        end
        if ~fault_d
            fault1time_d = cat(1,fault1time_d,'X')
            fault2time_d = cat(1,fault2time_d,'X')
        end
        if fault_t
            fault1time_t =
cat(1,fault1time_t,fault1timeonlyx_t(length(fault1timeonlyx_t),1));
            fault2time_t =
cat(1,fault2time_t,fault2timeonlyx_t(length(fault2timeonlyx_t),1));
            fault1timeonlyx_t =
removerows(fault1timeonlyx_t,length(fault1timeonlyx_t));
            fault2timeonlyx_t =
removerows(fault2timeonlyx_t,length(fault2timeonlyx_t));
        end
        if ~fault_t
            fault1time_t = cat(1,fault1time_t,'X')
            fault2time_t = cat(1,fault2time_t,'X')
        end
        if (fault_p && fault_e) || (fault_p && fault_d) || (fault_e &&
fault_d) || (fault_t && fault_p) || (fault_t && fault_e) || (fault_t &&
fault_d)
            timestore = time;
            time = timestore + brkcorr;     %wait until faulted system
has stabilized
```

```
        end
        if time < len           %as long as the result is before the
end of known time
            [wasgrowing1_e,wasgrowing2_e] = checkdir(rho1,rho2,time);

[rmin1_e,rmax1_e,rmin2_e,rmax2_e,thetashift1_e,thetashift2_e,wasgrowing
1_e,wasgrowing2_e,time,minrho1_e,highrho1_e,maxrho1_e,minrho2_e,highrho
2_e,maxrho2_e] =
minmaxrho(rho1,rho2,theta1,theta2,wasgrowing1_e,wasgrowing2_e,time,len,
lowallow_e,highallow_e,maxallow_e,ewaittimex,emaxtimex,quartertimex);
        end
        error1recent_p = 0;     % reset these variables to prepare for
the possibility of another fault
        error2recent_p = 0;
        error1recent_e = 0;
        error2recent_e = 0;
        error1recent_d = 0;
        error2recent_d = 0;
        temptime1_p = 0;
        temptime2_p = 0;
        temptime1_e = 0;
        temptime2_e = 0;
        temptheta1_p = 0;
        temptheta2_p = 0;
        temptheta1_e = 0;
        temptheta2_e = 0;
        fault_p = 0;
        fault_e = 0;
        fault_d = 0;
        faultwait_p = 0;
        faultwait_e = 0;
        dropbelow1_e = 0;
        dropbelow2_e = 0;
        dropbelow1_p = 0;
        dropbelow2_p = 0;
        delrho1triptime = 0;
        delrho2triptime = 0;
        delrhomax1 = 10.^9;
        delrhomax2 = 10.^9;
        delrhoallow = delrhoallowstart;
        misfaultcheck_d = 0;
        deltheta1triptime = 0;
        deltheta2triptime = 0;
        delthetamax1 = 0;
        delthetamax2 = 0;
        delthetamin1 = 0;
        delthetamin2 = 0;
    end

    time = time + 1;

end
```

```
%Fault analysis - after end of time loop


if fault == 0
    disp('No faults detected.')
end
if fault == 1
    tacc = 1;                              %Counter for while loop
    while tacc <= length (fault1time_p) %Improve accuracy of
calculation by using the
        if (fault1time_p(tacc,1) == 'X')%earliest available times
            if (fault1time_e(tacc,1) == 'X') %d
                newfault1time =
cat(1,newfault1time,fault1time_d(tacc,1));
            else
                if (fault1time_d(tacc,1) == 'X') %e
                    if (fault1time_t(tacc,1) == 'X')
                        fault1timeonly_e =
cat(1,fault1timeonly_e,fault1time_e(tacc,1));
                        fault1typeonly_e =
cat(1,fault1typeonly_e,fault1type_e(tacc,1));
                    else
                        newfault1time =
cat(1,newfault1time,fault1time_e(tacc,1));
                    end
                else %d&e
                    if (fault1time_e(tacc,1) < fault1time_d(tacc,1))
                        newfault1time =
cat(1,newfault1time,fault1time_e(tacc,1));
                    else
                        newfault1time =
cat(1,newfault1time,fault1time_d(tacc,1));
                    end
                end
            end
        else
            if (fault1time_e(tacc,1) == 'X')
                if (fault1time_d(tacc,1) == 'X') %p
                    if (fault1time_t(tacc,1) == 'X')
                        fault1timeonly_p =
cat(1,fault1timeonly_p,fault1time_p(tacc,1));
                        fault1typeonly_p =
cat(1,fault1typeonly_p,fault1type_p(tacc,1));
                    else
                        newfault1time =
cat(1,newfault1time,fault1time_p(tacc,1));
                    end
                else %d&p
                    if (fault1time_p(tacc,1) < fault1time_d(tacc,1))
                        newfault1time =
cat(1,newfault1time,fault1time_p(tacc,1));
                    else
                        newfault1time =
cat(1,newfault1time,fault1time_d(tacc,1));
                    end
                end
```

```
            else
                if (fault1time_d(tacc,1) == 'X') %e&p
                    if (fault1time_e(tacc,1) < fault1time_p(tacc,1))
                        newfault1time =
cat(1,newfault1time,fault1time_e(tacc,1));
                    else
                        newfault1time =
cat(1,newfault1time,fault1time_p(tacc,1));
                    end
                else %d&e&p
                    if (fault1time_p(tacc,1) <= fault1time_d(tacc,1))
&& (fault1time_p(tacc,1) <= fault1time_e(tacc,1))
                        newfault1time =
cat(1,newfault1time,fault1time_p(tacc,1));
                    elseif (fault1time_e(tacc,1) <=
fault1time_d(tacc,1)) && (fault1time_e(tacc,1) <= fault1time_p(tacc,1))
                        newfault1time =
cat(1,newfault1time,fault1time_e(tacc,1));
                    else
                        newfault1time =
cat(1,newfault1time,fault1time_d(tacc,1));
                    end
                end
            end
        end
        if (fault1time_t(tacc,1) ~= 'X') %include data from t if better
            if (fault1time_t(tacc,1) < newfault1time(tacc,1))
                newfault1time = removerows(newfault1time,tacc);
                newfault1time =
cat(1,newfault1time,fault1time_t(tacc,1));
            end
        end
        if (fault2time_p(tacc,1) == 'X')
            if (fault2time_e(tacc,1) == 'X') %d
                newfault2time =
cat(1,newfault2time,fault2time_d(tacc,1));
            else
                if (fault2time_d(tacc,1) == 'X') %e
                    if (fault2time_t(tacc,1) == 'X')
                        fault2timeonly_e =
cat(1,fault2timeonly_e,fault2time_e(tacc,1));
                        fault2typeonly_e =
cat(1,fault2typeonly_e,fault2type_e(tacc,1));
                    else
                        newfault2time =
cat(1,newfault2time,fault2time_e(tacc,1));
                    end
                else %d&e
                    if (fault2time_e(tacc,1) < fault2time_d(tacc,1))
                        newfault2time =
cat(1,newfault2time,fault2time_e(tacc,1));
                    else
                        newfault2time =
cat(1,newfault2time,fault2time_d(tacc,1));
                    end
                end
            end
```

```
        else
            if (fault2time_e(tacc,1) == 'X')
                if (fault2time_d(tacc,1) == 'X') %p
                    if (fault2time_t(tacc,1) == 'X')
                        fault2timeonly_p =
cat(1,fault2timeonly_p,fault2time_p(tacc,1));
                        fault2typeonly_p =
cat(1,fault2typeonly_p,fault2type_p(tacc,1));
                    else
                        newfault2time =
cat(1,newfault2time,fault2time_p(tacc,1));
                    end
                else %d&p
                    if (fault2time_p(tacc,1) < fault2time_d(tacc,1))
                        newfault2time =
cat(1,newfault2time,fault2time_p(tacc,1));
                    else
                        newfault2time =
cat(1,newfault2time,fault2time_d(tacc,1));
                    end
                end
            else
                if (fault2time_d(tacc,1) == 'X') %e&p
                    if (fault2time_e(tacc,1) < fault2time_p(tacc,1))
                        newfault2time =
cat(1,newfault2time,fault1time_e(tacc,1));
                    else
                        newfault2time =
cat(1,newfault2time,fault1time_p(tacc,1));
                    end
                else %d&e&p
                    if (fault2time_p(tacc,1) <= fault2time_d(tacc,1))
&& (fault2time_p(tacc,1) <= fault2time_e(tacc,1))
                        newfault2time =
cat(1,newfault2time,fault2time_p(tacc,1));
                    elseif (fault2time_e(tacc,1) <=
fault2time_d(tacc,1)) && (fault2time_e(tacc,1) <= fault2time_p(tacc,1))
                        newfault2time =
cat(1,newfault2time,fault2time_e(tacc,1));
                    else
                        newfault2time =
cat(1,newfault2time,fault2time_d(tacc,1));
                    end
                end
            end
        end
        if (fault2time_t(tacc,1) ~= 'X') %include data from t if better
            if (fault2time_t(tacc,1) < newfault2time(tacc,1))
                newfault2time = removerows(newfault2time,tacc);
                newfault2time =
cat(1,newfault2time,fault2time_t(tacc,1));
            end
        end
        tacc = tacc + 1;
    end

    fault1timeonly_d = cat(1,fault1timeonly_d,fault1timeonlyx_d);
```

```
    fault2timeonly_d = cat(1,fault2timeonly_d,fault2timeonlyx_d);
    fault1timeonly_t = cat(1,fault1timeonly_t,fault1timeonlyx_t);
    fault2timeonly_t = cat(1,fault2timeonly_t,fault2timeonlyx_t);

    [fault1type_p] = typechange(fault1type_p);
    [fault2type_p] = typechange(fault2type_p);
    [fault1type_e] = typechange(fault1type_e);
    [fault2type_e] = typechange(fault2type_e);
    [fault1typeonly_p] = typechange(fault1typeonly_p);
    [fault2typeonly_p] = typechange(fault2typeonly_p);
    [fault1typeonly_e] = typechange(fault1typeonly_e);
    [fault2typeonly_e] = typechange(fault2typeonly_e);

    fault1timecorr = newfault1time.*tstep./10^6; %fault times corrected
from matrix
    fault2timecorr = newfault2time.*tstep./10^6; %indices to actual
times

    fault1timeonlycorr_p = fault1timeonly_p.*tstep./10^6; %fault times
corrected from matrix
    fault1timeonlycorr_e = fault1timeonly_e.*tstep./10^6; %indices to
actual times
    fault1timeonlycorr_d = fault1timeonly_d.*tstep./10^6;
    fault1timeonlycorr_t = fault1timeonly_t.*tstep./10^6;
    fault2timeonlycorr_p = fault2timeonly_p.*tstep./10^6; %fault times
corrected from matrix
    fault2timeonlycorr_e = fault2timeonly_e.*tstep./10^6; %indices to
actual times
    fault2timeonlycorr_d = fault2timeonly_d.*tstep./10^6;
    fault2timeonlycorr_t = fault2timeonly_t.*tstep./10^6;

    len_from_gen = (totlength + vel*10^-3*(fault1timecorr-
fault2timecorr))/2;
    len_from_load = (totlength - len_from_gen);
    fault_time = (tstart + fault1timecorr - len_from_gen/vel);

    len_from_gen_only_e = (totlength + vel*10^-3*(fault1timeonlycorr_e-
fault2timeonlycorr_e))/2;
    len_from_load_only_e = (totlength - len_from_gen_only_e);
    fault_time_only_e = (tstart + fault1timeonlycorr_e -
len_from_gen_only_e/vel);
    len_from_gen_only_p = (totlength + vel*10^-3*(fault1timeonlycorr_p-
fault2timeonlycorr_p))/2;
    len_from_load_only_p = (totlength - len_from_gen_only_p);
    fault_time_only_p = (tstart + fault1timeonlycorr_p -
len_from_gen_only_p/vel);
    len_from_gen_only_d = (totlength + vel*10^-3*(fault1timeonlycorr_d-
fault2timeonlycorr_d))/2;
    len_from_load_only_d = (totlength - len_from_gen_only_d);
    fault_time_only_d = (tstart + fault1timeonlycorr_d -
len_from_gen_only_d/vel);
    len_from_gen_only_t = (totlength + vel*10^-3*(fault1timeonlycorr_t-
fault2timeonlycorr_t))/2;
    len_from_load_only_t = (totlength - len_from_gen_only_t);
    fault_time_only_t = (tstart + fault1timeonlycorr_t -
len_from_gen_only_t/vel);
```

```
    testcounter = 1;
    while (testcounter <= length(len_from_gen))
        if (len_from_gen(testcounter,1) < 0) ||
(len_from_load(testcounter,1) < 0) || (fault_time(testcounter,1) <= 0)
            len_from_gen = removerows(len_from_gen,testcounter);
            len_from_load = removerows(len_from_load,testcounter);
            timeneg = cat(1,timeneg,fault_time(testcounter,1));
            fault_time = removerows(fault_time,testcounter);
            negdistance = 1;
            testcounter = testcounter - 1;      %since a row will be
removed, want to test the same row # next time
        end
        testcounter = testcounter + 1;
    end
    testcounter = 1;
    while (testcounter <= length(len_from_gen_only_e))
        if (len_from_gen_only_e(testcounter,1) < 0) ||
(len_from_load_only_e(testcounter,1) < 0) ||
(fault_time_only_e(testcounter,1) <= 0)
            len_from_gen_only_e =
removerows(len_from_gen_only_e,testcounter);
            len_from_load_only_e =
removerows(len_from_load_only_e,testcounter);
            timeneg_e =
cat(1,timeneg_e,fault_time_only_e(testcounter,1));
            fault_time_only_e =
removerows(fault_time_only_e,testcounter);
            negdistance = 1;
            testcounter = testcounter - 1;      %since a row will be
removed, want to test the same row # next time
        end
        testcounter = testcounter + 1;
    end
    testcounter = 1;
    while (testcounter <= length(len_from_gen_only_p))
        if (len_from_gen_only_p(testcounter,1) < 0) ||
(len_from_load_only_p(testcounter,1) < 0) ||
(fault_time_only_p(testcounter,1) <= 0)
            len_from_gen_only_p =
removerows(len_from_gen_only_p,testcounter);
            len_from_load_only_p =
removerows(len_from_load_only_p,testcounter);
            timeneg_p =
cat(1,timeneg_p,fault_time_only_p(testcounter,1));
            fault_time_only_p =
removerows(fault_time_only_p,testcounter);
            negdistance = 1;
            testcounter = testcounter - 1;      %since a row will be
removed, want to test the same row # next time
        end
        testcounter = testcounter + 1;
    end
    testcounter = 1;
    while (testcounter <= length(len_from_gen_only_d))
```

```
        if (len_from_gen_only_d(testcounter,1) < 0) ||
(len_from_load_only_d(testcounter,1) < 0) ||
(fault_time_only_d(testcounter,1) <= 0)
            len_from_gen_only_d =
removerows(len_from_gen_only_d,testcounter);
            len_from_load_only_d =
removerows(len_from_load_only_d,testcounter);
            timeneg_d =
cat(1,timeneg_d,fault_time_only_d(testcounter,1));
            fault_time_only_d =
removerows(fault_time_only_d,testcounter);
            negdistance = 1;
            testcounter = testcounter - 1;      %since a row will be
removed, want to test the same row # next time
        end
        testcounter = testcounter + 1;
    end
    testcounter = 1;
    while (testcounter <= length(len_from_gen_only_t))
        if (len_from_gen_only_t(testcounter,1) < 0) ||
(len_from_load_only_t(testcounter,1) < 0) ||
(fault_time_only_t(testcounter,1) <= 0)
            len_from_gen_only_t =
removerows(len_from_gen_only_t,testcounter);
            len_from_load_only_t =
removerows(len_from_load_only_t,testcounter);
            timeneg_t =
cat(1,timeneg_t,fault_time_only_t(testcounter,1));
            fault_time_only_t =
removerows(fault_time_only_t,testcounter);
            negdistance = 1;
            testcounter = testcounter - 1;      %since a row will be
removed, want to test the same row # next time
        end
        testcounter = testcounter + 1;
    end

    if (length(len_from_gen) > 0)
        disp('Faults were found with at least two algorithms at the
following locations:')
        len_from_gen
        len_from_load
        disp('They occurred at the following relative times:')
        fault_time
        disp('The indications for these faults were as follows:')
        fault1type_p
        fault2type_p
        fault1type_e
        fault2type_e
    end
    if (length(len_from_gen_only_e) > 0)
        disp('Faults were found with just the ellipse algorithm at the
following locations:')
        len_from_gen_only_e
        len_from_load_only_e
        disp('They occurred at the following relative times:')
        fault_time_only_e
```

```
        disp('The indications for these faults were as follows:')
        fault1typeonly_e
        fault2typeonly_e
    end
    if (length(len_from_gen_only_p) > 0)
        disp('Faults were found with just the previous value algorithm
at the following locations:')
        len_from_gen_only_p
        len_from_load_only_p
        disp('They occurred at the following relative times:')
        fault_time_only_p
        disp('The indications for these faults were as follows:')
        fault1typeonly_p
        fault2typeonly_p
    end
    if (length(len_from_gen_only_d) > 0)
        disp('Faults were found with just the delta rho algorithm at
the following locations:')
        len_from_gen_only_d
        len_from_load_only_d
        disp('They occurred at the following relative times:')
        fault_time_only_d
        disp('Note that the delta rho algorithm finds high impedance
faults better than the other algorithms but is more likely to trip
accidentally due to noise or sudden changes to the system.')
    end
    if (length(len_from_gen_only_t) > 0)
        disp('Faults were found with just the delta theta algorithm at
the following locations:')
        len_from_gen_only_t
        len_from_load_only_t
        disp('They occurred at the following relative times:')
        fault_time_only_t
        disp('Note that the delta theta algorithm finds high impedance
faults better than the other algorithms but is more likely to trip
accidentally due to noise or sudden changes to the system.')
    end

end
if (negdistance == 1)
    disp('Warning: At least one fault calculation resulted in a
negative distance calculation. Note that associated times have a margin
of error of several milliseconds.')
    if (length(timeneg) > 0)
        disp('Times resulting in negative distances detected by both
algorithms:')
        timeneg
    end
    if (length(timeneg_e) > 0)
        disp('Times resulting in negative distances detected by the
expected ellipse algorithm:')
        timeneg_e
    end
    if (length(timeneg_p) > 0)
        disp('Times resulting in negative distances detected by the
previous value algorithm:')
        timeneg_p
```

```
    end
end

if (length(probtime1_p) > 0) || (length(probtime2_p) > 0)
    disp('Error: A fault has been detected at one end of the
transmission line using the previous value algorithm, but nothing has
indicated the fault at the other end. This occurred at the following
time:')
    if (length(probtime1_p) > 0)
        probtime1_p
    end
    if (length(probtime2_p) > 0)
        probtime2_p
    end
end
if (length(probtime1_e) > 0) || (length(probtime2_e) > 0)
    disp('Error: A fault has been detected at one end of the
transmission line using the expected ellipse algorithm, but nothing has
indicated the fault at the other end. This occurred at the following
time:')
    if (length(probtime1_e) > 0)
        probtime1_e
    end
    if (length(probtime2_e) > 0)
        probtime2_e
    end
end



figure(1)
polar(theta1,rho1);
title('Magnetic Field Plot at Generator End of Transmission Line.');

figure(2)
polar(theta2,rho2);
title('Magnetic Field Plot at Load End of Transmission Line.');

end









%Subfunctions that are called in the main function

function
[rhomin_i1,rhomax_i1,rhomin_i2,rhomax_i2,thetashift_i1,thetashift_i2,gr
owing_i1,growing_i2,time_i,minrho_i1,highrho_i1,maxrho_i1,minrho_i2,hig
hrho_i2,maxrho_i2] =
```

```
minmaxrho(rho_i1,rho_i2,theta_i1,theta_i2,growing_i1,growing_i2,time_i,
len_i,lowallow_i,highallow_i,maxallow_i,ewaittimex_i,emaxtimex_i,quarte
rtimex_i)
%minmaxrho sets/resets the maximum and minimum rho values seen by the
%   sensors on both ends of the transmission line. This information is
%   later used in determining whether or not a fault is occurring/has
%   occurred.

rhomin_i1 = 0;
rhomax_i1 = 0;
rhomin_i2 = 0;
rhomax_i2 = 0;
thetashift_i1 = 0;
thetashift_i2 = 0;
rhomintest_i1 = 0;
rhomaxtest_i1 = 0;
rhomintest_i2 = 0;
rhomaxtest_i2 = 0;
numreps = 1;
lastrmintime1_i = 0;
lastrmintime2_i = 0;
lastrmaxtime1_i = 0;
lastrmaxtime2_i = 0;

while (((rhomintest_i1 < numreps) || (rhomaxtest_i1 < numreps) ||
(rhomintest_i2 < numreps) || (rhomaxtest_i2 < numreps)) ||
(rho_i1(time_i,1) > rhomax_i1) && (rho_i2(time_i,1) > rhomax_i2) &&
(rho_i2(time_i,1) < rhomin_i2) && (rho_i2(time_i,1) < rhomin_i2)) &&
(time_i < len_i)
    if ((rho_i1((time_i-ewaittimex_i),1) > rho_i1(time_i,1)) &&
growing_i1==1) && (time_i > (lastrmintime1_i + emaxtimex_i))
        timemaxcheck_i = 0;
        rhomax_i1 = 0;
        while (timemaxcheck_i < quartertimex_i) && ((time_i-
timemaxcheck_i) > 0)
            if (rho_i1((time_i-timemaxcheck_i),1) > rhomax_i1)
                rhomax_i1 = rho_i1((time_i-timemaxcheck_i),1);
                thetashift_i1 = theta_i1((time_i-timemaxcheck_i),1);
            end
            timemaxcheck_i = timemaxcheck_i + 1;
        end
        lastrmaxtime1_i = time_i;
        growing_i1 = 0;
        rhomaxtest_i1 = rhomaxtest_i1 + 1;
    end
    if ((rho_i2((time_i-ewaittimex_i),1) > rho_i2(time_i,1)) &&
growing_i2==1) && (time_i > (lastrmintime2_i + emaxtimex_i))
        timemaxcheck_i = 0;
        rhomax_i2 = 0;
        while (timemaxcheck_i < quartertimex_i) && ((time_i-
timemaxcheck_i) > 0)
            if (rho_i2((time_i-timemaxcheck_i),1) > rhomax_i2)
                rhomax_i2 = rho_i2((time_i-timemaxcheck_i),1);
                thetashift_i2 = theta_i2((time_i-timemaxcheck_i),1);
            end
            timemaxcheck_i = timemaxcheck_i + 1;
```

```
        end
        lastrmaxtime2_i = time_i;
        growing_i2 = 0;
        rhomaxtest_i2 = rhomaxtest_i2 + 1;
    end
    if ((rho_i1((time_i-ewaittimex_i),1) < rho_i1(time_i,1)) &&
growing_i1==0) && (time_i > (lastrmaxtime1_i + emaxtimex_i))
        timemaxcheck_i = 0;
        rhomin_i1 = 10.^9;
        while (timemaxcheck_i < quartertimex_i) && ((time_i-
timemaxcheck_i) > 0)
            if (rho_i1((time_i-timemaxcheck_i),1) < rhomin_i1)
                rhomin_i1 = rho_i1((time_i-timemaxcheck_i),1);
            end
            timemaxcheck_i = timemaxcheck_i + 1;
        end
        lastrmintime1_i = time_i;
        growing_i1 = 1;
        if rhomin_i1 < 0.1
            rhomin_i1 = 0.1;
        end
        rhomintest_i1 = rhomintest_i1 + 1;
    end
    if ((rho_i2((time_i-ewaittimex_i),1) < rho_i2(time_i,1)) &&
growing_i2==0) && (time_i > (lastrmaxtime2_i + emaxtimex_i))
        timemaxcheck_i = 0;
        rhomin_i2 = 10.^9;
        while (timemaxcheck_i < quartertimex_i) && ((time_i-
timemaxcheck_i) > 0)
            if (rho_i2((time_i-timemaxcheck_i),1) < rhomin_i2)
                rhomin_i2 = rho_i2((time_i-timemaxcheck_i),1);
            end
            timemaxcheck_i = timemaxcheck_i + 1;
        end
        lastrmintime2_i = time_i;
        growing_i2 = 1;
        if rhomin_i2 < 0.1
            rhomin_i2 = 0.1;
        end
        rhomintest_i2 = rhomintest_i2 + 1;
    end
    time_i = time_i + 1;
end

predrho_i1 = (rhomax_i1.*rhomin_i1)./((rhomax_i1.*sin(theta_i1-
thetashift_i1)).^2+(rhomin_i1.*cos(theta_i1-
thetashift_i1)).^2+0.0001).^(1/2);
predrho_i2 = (rhomax_i2.*rhomin_i2)./((rhomax_i2.*sin(theta_i2-
thetashift_i2)).^2+(rhomin_i2.*cos(theta_i2-
thetashift_i2)).^2+0.0001).^(1/2);

if (rhomin_i1 == 0.1)
    minrho_i1 = zeros(len_i,1);
else
    minrho_i1 = predrho_i1.*(1-lowallow_i);
end
```

```
if (rhomin_i2 == 0.1)
    minrho_i2 = zeros(len_i,1);
else
    minrho_i2 = predrho_i2.*(1-lowallow_i);
end
highrho_i1 = predrho_i1.*(1+highallow_i);
highrho_i2 = predrho_i2.*(1+highallow_i);
maxrho_i1 = predrho_i1.*(1+maxallow_i);
maxrho_i2 = predrho_i2.*(1+maxallow_i);

end



function [growing_c1,growing_c2] = checkdir(rho_c1,rho_c2,time_c)
%checkdir checks the whether rho_c1 and rho_c2 are growing or not and
%   returns this information in the variables growing_c1 and growing_c2
%   (1 = rho is increasing, 0 = rho is decreasing)

if (rho_c1((time_c-1),1) > rho_c1(time_c,1))%initialize direction of
movement
    growing_c1 = 0;
else
    growing_c1 = 1;
end
if (rho_c2((time_c-1),1) > rho_c2(time_c,1))
    growing_c2 = 0;
else
    growing_c2 = 1;
end

end



function [faulttime_to] = typechange(faulttime_ti)
%typechange changes the numerical representations of the reasons for
fault
%detection into a string describing the result

faulttime_to = [];
ttype = 1;
while ttype <= length(faulttime_ti)
    if faulttime_ti(ttype,1) == 1
        faulttime_to = strvcat(faulttime_to,'Increase above maximum
allowed value');
    elseif faulttime_ti(ttype,1) == 2
        faulttime_to = strvcat(faulttime_to,'Decrease below minimum
then unexpected increase');
    elseif faulttime_ti(ttype,1) == 3
        faulttime_to = strvcat(faulttime_to,'Increase above high
allowed value');
    elseif faulttime_ti(ttype,1) == 4
        faulttime_to = strvcat(faulttime_to,'Decrease below minimum
allowed value');
```

```
    elseif faulttime_ti(ttype,1) == 'X'
        faulttime_to = strvcat(faulttime_to,'No fault detected, or two
other algorithms detected the fault before this algorithm');
    else
        faulttime_to = strvcat(faulttime_to,'Unknown result');
    end
    ttype = ttype + 1;
end

end
```